January 1999

Path Constraints for Databases With or Without Schemas

Wenfei Fan
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-99-04.

Path Constraints for Databases With or Without Schemas

Abstract
This dissertation introduces a path constraint language and investigates its associated implication and finite implication problems.

This path constraint language has proven useful in a variety of database contexts, ranging from semistructured data as found for instance on the Web, to structured data such as data in object-oriented databases. It is capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization.

Path constraint implication is investigated for two models for semistructured data: the semistructured data model and the deterministic data model. Databases in these models are unconstrained by any type system or schema. For the semistructured data model, it is shown that, despite the simple syntax of the constraint language, its associated implication problem is r.e. complete and its finite implication problem is co-r.e. complete. However, in light of these undecidability results, several decidable fragments of the constraint language are identified. These fragments suffice to express many important integrity constraints such as referential integrity, inverse relationships and local database constraints. For the deterministic data model, it is shown that the implication and finite implication problems for the path constraint language are finitely axiomatizable and decidable in cubic-time. Path constraint implication is also studied for structured data, i.e., data constrained by a schema. In the context of three practical object-oriented data models, a number of complexity results on the implication and finite implication problems for the path constraint language are established. In addition, the interaction between path constraints and type systems is investigated. It is demonstrated that adding a type to the data may in some cases simplify the analysis of path constraint implication, and in other cases make it harder.

More specifically, it is shown that there is a path constraint implication problem that is decidable in PTIME in the untyped context, but that becomes undecidable when a type system is added. On the other hand, there is an implication problem that is undecidable in the untyped context, but becomes not only decidable in cubic-time but also finitely axiomatizable when a type system is imposed.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-99-04.

PATH CONSTRAINTS FOR DATABASES WITH OR WITHOUT SCHEMAS

Wenfei Fan

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1999

Peter Buneman
Supervisor of Dissertation

Scott Weinstein
Supervisor of Dissertation

Jean Gallier
Graduate Group Chairperson

To my parents.

ACKNOWLEDGMENTS

I am indebted to Peter Buneman and Scott Weinstein, my advisors. Peter made it possible for me to enroll in the Ph.D. program at Penn, introduced me to database theory, determined the theme of this dissertation, and provided invaluable help and support. This dissertation is profoundly marked by his scholarship and advice. Scott helped me to go through the most difficult time of my life. Without his kind encouragement and unfailing support, this dissertation would not have been possible. In addition, his deep insight into constraint theory and crucial advice on a number of proofs made direct contributions to this work. I am lucky to have had two extremely supportive advisors.

I am very grateful to Susan Davidson and Val Tannen for their help, support, advice, suggestions and encouragement. My heartfelt thanks also go to Leonid Libkin, Dan Suciu, Victor Vianu and Limsoon Wong for their valuable suggestions and comments. I am also grateful to Rona Machlin for her help, especially in the preparation of this dissertation.

I would like to thank the members of my thesis committee for their insightful comments, suggestions, questions and criticisms: Susan Davidson, Neil Immerman (University of Massachusetts), Sampath Kannan, Mark Steedman and Val Tannen. This work has also benefited greatly from the discussions at the "Tuesday Database Club", whose members include Peter Buneman, Susan Davidson, Alin Deutsch, Carmem Hara, Scott Harker, Gerd Hillebrand, Anthony Kosky, Zoe Lacroix, Leonid Libkin, Hartmut Liefke, Yi Luo, Rona Machlin, Lucian Popa, Arnaud Sahuguet, Dan Suciu, Wang-Chiew Tan, Val Tannen, and Limsoon Wong.

My special thanks go to Jean Gallier, Carl A. Gunter, Val Tannen and Scott Weinstein for serving on my WPE II committee.

I would also like to thank the members of the computational biological group, in particular, Chris Overton and Kyle Hart, for their collaboration on my study of text databases and in maintaining Kleisli.

I have also received great help from the administrative staff in the CIS department. In particular, I would like to thank Michael Felker. I kept bothering him from the first day I

came to Penn and he always helped me out.

I am deeply grateful to the Institute for Research in Cognitive Science, for providing me a graduate fellowship during the dissertation research.

Finally, I would like to thank my family: my parents for allowing me to study abroad; my wife for her love, devotion, support and patience; and my sisters for looking after my parents during my absence from home. My deep gratitude also goes to my friends, whom I consider part of my family, for their continuous support: Lois and Bill Brady, Hao Dai, Tsang Chan, Jade and Alan Kaiser, Marvin and Tabasom Khoshkhassal, Wa Ngai, Yedong Sun and Dianping Yang.

ABSTRACT

PATH CONSTRAINTS FOR DATABASES WITH OR WITHOUT SCHEMAS

Wenfei Fan

Advisors: Peter Buneman and Scott Weinstein

This dissertation introduces a path constraint language and investigates its associated implication and finite implication problems.

This path constraint language has proven useful in a variety of database contexts, ranging from semistructured data as found for instance on the Web, to structured data such as data in object-oriented databases. It is capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization.

Path constraint implication is investigated for two models for semistructured data: the semistructured data model and the deterministic data model. Databases in these models are unconstrained by any type system or schema. For the semistructured data model, it is shown that, despite the simple syntax of the constraint language, its associated implication problem is r.e. complete and its finite implication problem is co-r.e. complete. However, in light of these undecidability results, several decidable fragments of the constraint language are identified. These fragments suffice to express many important integrity constraints such as referential integrity, inverse relationships and local database constraints. For the deterministic data model, it is shown that the implication and finite implication problems for the path constraint language are finitely axiomatizable and decidable in cubic-time.

Path constraint implication is also studied for structured data, i.e., data constrained by a schema. In the context of three practical object-oriented data models, a number of complexity results on the implication and finite implication problems for the path constraint language are established. In addition, the interaction between path constraints and type systems is investigated. It is demonstrated that adding a type to the data may in some cases simplify the analysis of path constraint implication, and in other cases make it harder.

More specifically, it is shown that there is a path constraint implication problem that is decidable in PTIME in the untyped context, but that becomes undecidable when a type system is added. On the other hand, there is an implication problem that is undecidable in the untyped context, but becomes not only decidable in cubic-time but also finitely axiomatizable when a type system is imposed.

Contents

List of Tables

List of Figures

Part I

Introduction

This dissertation investigates a class of path constraints and their associated implication and finite implication problems in a variety of database contexts, ranging from semistructured data, i.e., data whose structure is not constrained by a schema, to structured data, i.e., data constrained by a schema.

There is a natural analogy between this work and inclusion dependency theory established for the relational data model. The theory of inclusion dependencies constitutes an important part of relational database theory. It provides a formal mechanism for expressing certain integrity constraints that commonly arise in practice, and has proven useful for optimizing queries and for performing updates in the context of relational databases (see [5] for more detailed discussions of inclusion dependency theory). In the same way, the class of path constraints investigated here is important to a variety of data models developed to meet the demands of a wide range of applications. At one end of the spectrum, these models include sophisticated ones with richer constructs than the relational model, such as object-oriented models proposed to deal with, for example, computer-aided design and office information systems. At the other end, among these models are semistructured data models, which have recently emerged in response to the needs of data integration, data browsing and querying unstructured data as found for instance on the Web. In these models, these path constraints are capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization.

To use path constraints in query optimization, it is important to be able to reason about them. That is, one needs to settle the question of constraint implication. The implication and finite implication problems for path constraints, which are the central technical

problems investigated in this dissertation, are also interesting in their own right. They are connected to the key issues of computer sciences. In addition, these problems lie at the confluence of finite model theory, bounded variable logics, feature logics, logics of programs and language theory. The complexity results established here may shed light on these related topics.

Part I of the dissertation introduces the path constraint language. More specifically,

- Chapter 1 presents the motivation for the study of the path constraints, briefly describes the main results of the dissertation, covers related research, and provides a road map through this dissertation.
- Chapter 2 formally defines the path constraint language, and describes its associated axiomatizability, implication and finite implication problems.

Chapter 1

Introduction

This chapter presents the motivation for studying path constraints (Section 1.1), describes the main results of this dissertation (Section 1.2), and covers related work (Section 1.3). The structure of the dissertation is outlined in Section 1.4.

1.1 Motivation

The fundamental motivation for the study of path constraints is to incorporate more semantics into data models, including those with richer constructs than the relational model. To illustrate this, consider the following object-oriented schema:

```
class student{
      Name:     string;
      Taking:   set(course);
}
class course{
      CName:     string;
      Enrolled:  set(student);
}

Students:  set(student);
Courses:   set(course);
```

in which we assume that the declarations `Students` and `Courses` define (persistent) entry points into the database. As it stands, this declaration does not provide full information about the intended structure. Given such a database we would expect the following

informally stated constraints to hold:

(a) $\forall s \in Students \ \forall c \in s.Taking \ (c \in Courses)$

(b) $\forall c \in Courses \ \forall s \in c.Enrolled \ (s \in Students)$

That is, any course taken by a student must be a course that occurs in the database extent of courses, and any student enrolled in a course must be a student that similarly occurs in the database. We shall call such constraints *extent* constraints.

We might also expect an *inverse relationship* to hold between `Taking` and `Enrolled`. Object-oriented databases differ in the ways they enable one to state and enforce extent constraints and inverse relationships. Compare, for example, O_2 [11] and ObjectStore [59].

Let us develop a more formal notation for describing such constraints. To do this we borrow an idea that has been exploited in semistructured data models (e.g., [8, 20, 68, 69, 70]) of regarding semistructured data as an edge-labeled graph. In our object-oriented database there are two sets of objects, `Students` and `Courses`. We express this in semistructured data representation by building a graph with a root node r and a node for each object. Edges connect the root to these object nodes, and these edges are labeled either `Students` or `Courses`. Edges emanating from these nodes indicate attributes or relationships with other objects and are appropriately labeled. For example, a node representing a student object has a single `Name` edge connected to a string node, and multiple `Taking` edges connected to course nodes. See Figure 1.1 for an example of such a graph.

Using this representation of data we can examine certain kinds of constraints.

Extent constraints. By taking edge labels as binary predicates, constraints of the form (a) and (b) above can be stated as:

$$\forall c \ (\exists s \ (Students(r, s) \wedge Taking(s, c)) \rightarrow Courses(r, c))$$
$$\forall s \ (\exists c \ (Courses(r, c) \wedge Enrolled(c, s)) \rightarrow Students(r, s))$$

Here r is a constant denoting the root node, and variables c, s range over vertices. The first constraint above states that any vertex that is reached from the root by following a `Students` edge followed by a `Taking` edge can also be reached from the root by following a

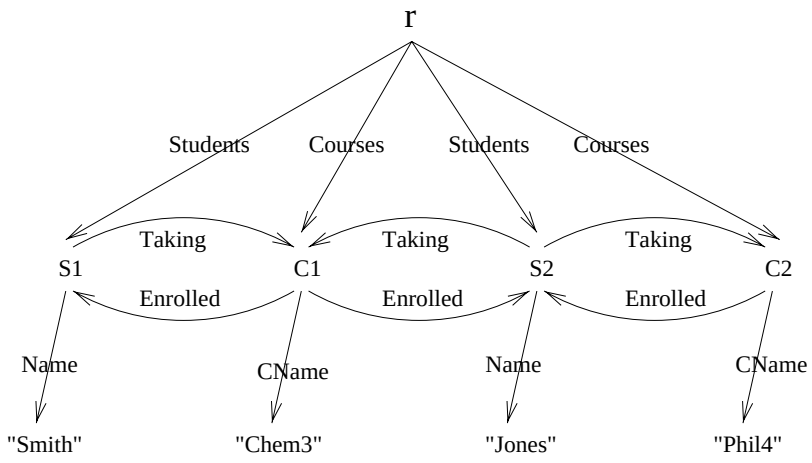

Figure 1.1: Representation of a student/course database

`Courses` edge. Similarly, the second asserts that any vertex that is reached from the root by following a `Courses` edge followed by an `Enrolled` edge can also be reached from the root by following a `Students` edge.

Inverse constraints. These are common in object-oriented databases [31]. With respect to our student/course schema, the inverse between `Taking` and `Enrolled` is expressed as:

$$\forall s\ (Students(r, s) \rightarrow \forall c\ (Taking(s, c) \rightarrow Enrolled(c, s)))$$
$$\forall c\ (Courses(r, c) \rightarrow \forall s\ (Enrolled(c, s) \rightarrow Taking(s, c)))$$

Local database constraints. In database integration it is sometimes desirable to make one database a component of another database, or to build a "database of databases". Suppose, for example, we want to bring together a number of student/course databases as described above. We might write something like:

```
class School-DB{
   DB-identifier:  string;
   Students:set(student);  // as defined above
   Courses: set(course);   // as defined above
}
Schools:  set(School-DB);
```

Now we may want certain constraints to hold on components of this database. For exam-

ple, the "extent constraints" described above now hold on each member of the `Schools` set. Here we refer to a component database such as a member of the set `Schools` as a *local database* and its constraints as *local database constraints*. Extending our graph representation by adding `Schools` edges from the new root node to the roots of local databases, the local extent constraints are:

$$\forall d \ \ (Schools(r,d) \rightarrow \forall c \ (\exists s \ (Students(d,s) \wedge Taking(s,c)) \rightarrow Courses(d,c)))$$
$$\forall d \ \ (Schools(r,d) \rightarrow \forall s \ (\exists c \ (Courses(d,c) \wedge Enrolled(c,s)) \rightarrow Students(d,s)))$$

The semantics conveyed by the constraints given above is important both as part of the database definition and also for query optimization. However, data models including widely used commercial object-oriented models cannot represent (at least part of) this semantic information. It is desirable to overcome this limitation by incorporating these constraints.

A path constraint language. These considerations give rise to the question whether there is a constraint language which will capture these integrity constraints. To answer this question, we consider a class of constraints of either the form

$$\forall x \ (\alpha(r,x) \rightarrow \forall y \ (\beta(x,y) \rightarrow \gamma(x,y))),$$

or the form

$$\forall x \ (\alpha(r,x) \rightarrow \forall y \ (\beta(x,y) \rightarrow \gamma(y,x))),$$

where $\alpha(x,y)$ ($\beta(x,y)$, $\gamma(x,y)$) represents a path, i.e., a sequence of edge labels, from node x to node y. As demonstrated above, $\alpha(x,y)$ can be expressed as a logic formula with two free variables x and y by treating edge labels as binary predicates. These constraints are called *path constraints*. We use P_c to denote this class of path constraints. This path constraint language is capable of expressing all the integrity constraints we have so far encountered. It properly contains the class of word constraints introduced in [9].

This path constraint language is defined for a graph model in which data is represented as an edge-labeled rooted directed graph. In a graph representing a database, the root node r indicates a persistent entry point into the database, the vertices represent data entities, and the edges are labeled with attribute names.

Semistructured data can be represented in this graph model. Semistructured data is characterized as having no type constraints, irregular structure and missing schema [2, 18]. In short, it refers to data whose structure is not constrained by a schema. Semistructured data is commonly found on the World-Wide Web, in biological databases and after data integration. In particular, documents of XML (eXtensible Markup Language, [17, 76]) can also be viewed as semistructured data [38]. The unifying idea in modeling semistructured data is the representation of data as a graph in the graph model. For example, Figure 1.2 shows a graph representing the following XML data:

```
<?XML version = "1.0">
<bib>
     <book ISBN = "12"   author = "345">
           <title> ...  </title>
           <publisher> ...  </publisher>
     </book>
     <book ISBN = "23"   author = "123"   ref = "12">
           <title> ...  </title>
     </book>
     <book ISBN = "34"   author = "123"   ref = "23">
           <title> ...  </title>
     </book>

     <person SSN = "123"   wrote = "34 23">
            <name> ...  </name>
     </person>
     <person SSN = "345"   wrote = "12">
            <name> ...  </name>
            <age> ...  </age>
     </person>
</bib>
```

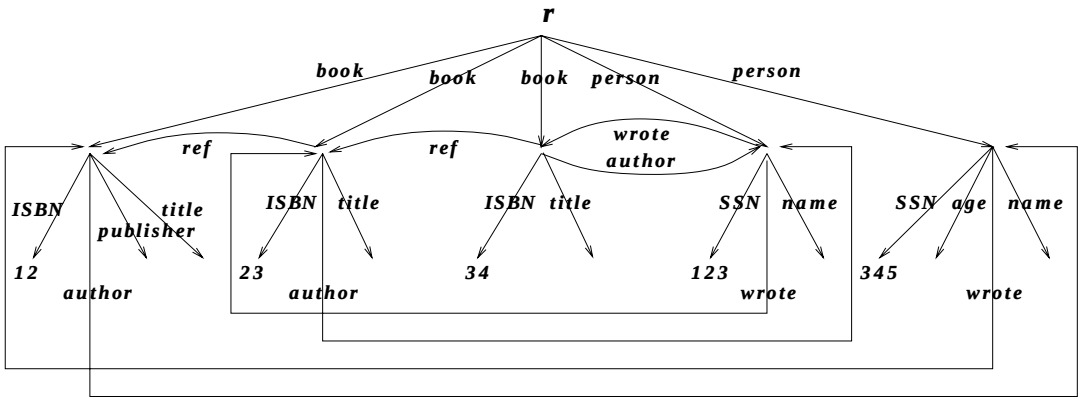

Figure 1.2: Representation of an XML document

With the recent popularity of semistructured data, this graph model is often referred to as the *semistructured data model* (abbreviated to $\mathcal{SM}$) in the literature [2, 18].

As illustrated by the student/course example, structured data can also be represented as a graph in $\mathcal{SM}$. Structured data is data constrained by a schema, such as in object-oriented databases. The type system or schema definition can be viewed as imposing a type constraint on the data. A graph representing the data must satisfy the type constraint.

Because of the expressive power of the graph model, path constraints of P_c are defined on both semistructured data and structured data. In these database contexts, path constraints have found a wide range of applications, which we next briefly describe.

Query optimization. Path constraints of P_c are useful in optimizing queries. To illustrate this, consider the student/course database given in Figure 1.1. Suppose, for example, we want to find the names of all the courses enrolled by students who are taking the course "Chem3". Without the inverse and extent constraints for the database described above, one would write the query as Q_1 (in OQL syntax [31]):

Q_1

```
select distinct c.CName
from   Courses c,
       c.Enrolled s,
       s.Taking c'
where  c'.CName = "Chem3"
```

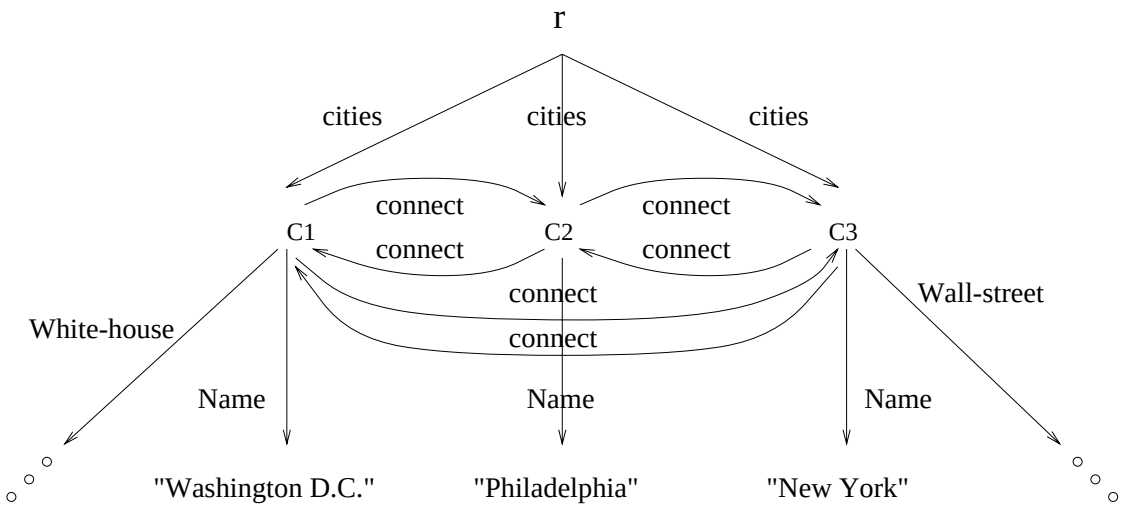

Figure 1.3: An example tour database

Given the inverse and extent constraints, one can show that Q_1 is equivalent to Q_2 given below:

Q_2

```
select distinct c.CName
from   Courses c',
       c'.Enrolled s,
       s.Taking c
where  c'.CName = "Chem3"
```

In other words, given the path constraints, one can rewrite Q_1 to Q_2. In most cases, Q_2 is more efficient than Q_1. Indeed, Q_2 complies with the familiar optimization principle originating in relational database theory: performing selections as early as possible.

As another example, consider the tour database given in Figure 1.3. Suppose we want to find all the cities connected to Philadelphia via one or more `connect` edges. Without knowledge of any path constraint on the database, the only way to formulate this query is by using some form of recursion. However, given the P_c constraints below:

$$\forall x \ \ (cities(r, x) \rightarrow \forall y \, (\exists z \ (connect(x, z) \wedge connect(z, y)) \rightarrow connect(x, y)))$$
$$\forall x \ \ (cities(r, x) \rightarrow \forall y \, (connect(x, y) \rightarrow connect(y, x)))$$

we are able to write the query without using recursion as (in Lorel [8] syntax):

```
select c
from   r.cities p, p.connect c
where  p.Name = "Philadelphia"
```

Semantic specification. Path constraints of P_c also offer a mechanism to describe structural information missing in the semistructured data model. Structural information is useful for query formulation and optimization. It also facilitates browsing data and exploring data formats especially when the data is exported to external applications. For XML documents, in particular, there are cases in which structural information is required to enforce content models. For this purpose, XML provides an optional document-structure grammar, called *DTD* (Document Type Descriptor. See [17, 76]), to describe semantic information. For example, consider the XML document represented in Figure 1.2. To ensure that cross-references in the document are handled properly, the following DTD is needed:

```
<!ELEMENT book   (title, publisher?)>
<!ATTLIST book
          ISBN      ID       #required
          author    IDREFs   #implied
          ref       IDREFs   #implied>

<!ELEMENT person   (name, age?)>
<!ATTLIST person
          SSN      ID       #required
          wrote    IDREFs   #implied>
```

This DTD specifies that a `book` element contains an element `title` and an optional element `publisher`. It has an `ISBN` attribute which is the key of the `book` element. It may also have `author` and `ref` attributes which refer to other elements of the document. Similarly, a `person` element contains a `name` element and an optional `age` element. It has a `SSN` attribute which is the key of the `person` element, and may contain a `wrote` attribute referring to other elements. However, XML and XML DTD are not sufficient to specify content

or semantics. To overcome this limitation, numerous object-oriented type systems have been proposed for adding structure or semantics to XML documents. Among these proposals, several [16, 44, 60, 61] advocate the need for integrity constraints. These integrity constraints are important in interpreting XML documents. Whether such constraints will be specified as extensions to existing type systems such as XML-Data [61], SOX [44], DCD [16], or whether they will be added as independent constructs, is not yet clear, and, in all probability, they will be added in both ways. For example, XLink [63] is independent of any type system and can express simple co-reference constraints. XML-Data, on the other hand, embeds restricted forms of extent and inverse constraints in it:

- Range constraints that declare restriction on the types of the elements to which an element may refer. For example, the `author` attribute of a `book` element can only refer to `person` elements, and the `wrote` of attribute of a `person` element can only refer to `book` elements. In XML-Data, these can be expressed as:

```
<elementType  id = "book">
     ...
     <attribute name = "author"  dt = "IDREFs"  range = "#person" />
</elementType>

<elementType  id = "person">
     ...
     <attribute name = "wrote"  dt = "IDREFs"  range = "#book" />
</elementType>
```

  It should be mentioned that these are just type restrictions and have no connection with the notion of extent.

- Correlatives that describe XML elements representing bidirectional relations. However, inverse relationships between attributes cannot be expressed as correlatives. For example, the inverse relationship between `author` attributes of `book` elements and `wrote` attributes of `person` elements cannot be expressed in XML-Data. If `author`

and `wrote` were XML elements rather than attributes, then the inverse relationships could be described as correlatives:

```
<elementType  id = "wrote">
     ...
     <correlative  type = "#author"/>
</elementType>
<elementType  id = "author">
     ...
     <correlative  type = "#wrote"/>
</elementType>
```

These integrity constraints can be expressed as P_c constraints. For example, on the XML document shown in Figure 1.2, one may impose the following extent and inverse constraints:

$$\forall\, x\, (\exists\, y\, (book(r,\ y) \wedge author(y,\ x)) \rightarrow person(r,\ x))$$
$$\forall\, x\, (\exists\, y\, (person(r,\ y) \wedge wrote(y,\ x)) \rightarrow book(r,\ x))$$
$$\forall\, x\, (\exists\, y\, (book(r,\ y) \wedge ref(y,\ x)) \rightarrow book(r,\ x))$$

$$\forall\, x\, (book(r,\ x) \rightarrow \forall\, y\, (author(x,\ y) \rightarrow wrote(y, x)))$$
$$\forall\, x\, (person(r,\ x) \rightarrow \forall\, y\, (wrote(x,\ y) \rightarrow author(y,\ x)))$$

Extent and inverse constraints on semistructured databases convey semantics commonly found in object-oriented databases. As another example, consider the following constraints for a Web database of a school:

$$\forall\, x\, (Dept(r, x) \rightarrow \forall\, y\, (TA(x, y) \rightarrow Student(x, y)))$$
$$\forall\, x\, (Dept(r, x) \rightarrow \forall\, y\, (TA(x, y) \rightarrow Employee(x, y)))$$
$$\forall\, x\, (Dept(r, x) \rightarrow \forall\, y\, ((Student(x, y) \wedge Employee(x, y)) \rightarrow TA(x, y))) \qquad (\dagger)$$

Here r indicates the home page of the school, which has links to the home pages of departments in the school. The home page of a department is in turn linked to home pages of employees, students and teaching assistants of the department. Abusing object-oriented database terms, these constraints state that

- TA of a department is a "subclass" of both Student and Employee of the department; and
- the "extent" of TA is the intersection of the "extents" of Student and Employee.

The first two constraints above are in P_c and the constraint (†) is in $P_c^{\wedge}$, which is a mild generalization of P_c to be studied in Chapter 5.

In the context of structured databases, it turns out that path constraints are also needed for many other reasons besides query optimization. First, many referential integrity constraints can be expressed as path constraints. For structured data, checking and maintaining these referential integrity constraints are central to performing updates and loading databases. Second, these referential integrity constraints are also used in database integration and transformations both to ensure information capacity preservation and to improve performance (see Chapter 5 of [58] for detailed discussions of this subject). Third, some fundamental features of object-oriented databases can be captured by path constraints. Including these constraints in new data models helps incorporate object-oriented features into these models.

Path constraint implication. To take advantage of path constraints, it is important to be able to reason about them. This gives rise to the question of logical implication for path constraints, the most important theoretical question in connection with path constraints. In general, we may know that a set of path constraints is satisfied by a database. The question of logical implication is: What other path constraints are necessarily satisfied by the database? To see why logical implication is important, consider the queries Q_1 and Q_2 given above against the student/course database. To show that Q_1 can be rewritten to Q_2, the following constraints of P_c are also needed in addition to the given inverse and extent constraints:

$$\forall s\ (\exists c_1\ (Courses(r, c_1) \wedge Enrolled(c_1, s)) \rightarrow \forall c\ (Taking(s, c) \rightarrow Enrolled(c, s)))$$
$$\forall c\ (\exists c_1\ (Courses(r, c_1) \wedge \exists s\ (Enrolled(c_1, s) \wedge Taking(s, c))) \rightarrow Courses(r, c))$$

To use these constraints, we need to show that they necessarily hold if the given extent and

inverse constraints hold. That is, they are implied by the given constraints. As another example, consider the XML document depicted in Figure 1.2. The path constraints below, which are implied by the given extent and inverse constraints on the data, are useful for understanding and querying the XML data:

$$\forall x\,(\exists y\,(book(r,\,y) \wedge \exists z\,(ref(y,\,z) \wedge ref(z,x))) \rightarrow book(r,\,x))$$
$$\forall x\,(\exists y\,(person(r,\,y) \wedge \exists z\,(wrote(y,\,z) \wedge \exists w\,(ref(z,w) \wedge author(w,x)))) \rightarrow person(r,\,x))$$
$$\forall x\,(\exists z\,(book(r,\,z) \wedge ref(z,x)) \rightarrow \forall y\,(author(x,\,y) \rightarrow wrote(y,x)))$$
$$\forall x\,(\exists z\,(book(r,\,z) \wedge \exists w\,(ref(z,\,w) \wedge author(w,x))) \rightarrow \forall y\,(wrote(x,\,y) \rightarrow author(y,\,x)))$$

As pointed out by [60], it is important to be able to reason about constraints on XML data. Because of this, a reasoning mechanism is expected to be built on top of RDF (Resource Description Framework [60]), which is a model proposed for specifying semantics of XML data.

There are two forms of implication problems associated with path constraints. In the context of structured data, databases are usually considered to be finite. Logical implication is called *finite implication* for the case in which only finite database instances are permitted. It is also interesting to consider logical implication in the traditional logic framework in which infinite instances are also allowed. In fact, in the context of semistructured data, some databases, e.g., the one consisting of all the pages on the Web, are sometimes viewed as infinite [10]. Logical implication is called *unrestricted implication*, or simply *implication*, for the case in which both finite database instances and infinite instances are permitted.

The central technical problems investigated in this dissertation are the implication and finite implication problems for path constraints. These problems are studied in a variety of database contexts.

Path constraint implication in the context of semistructured data. We investigate path constraint implication for two models for semistructured data. One is $\mathcal{SM}$ given above. The other, recently developed in [21] and referred to as *the deterministic data model* (abbreviated to $\mathcal{DM}$), is a variant of $\mathcal{SM}$ in which the edge relations of a graph are deterministic. That is, the edges emanating from any node in the graph have distinct labels.

In many applications, the deterministic data model is more appropriate for representing semistructured data. For example, when modeling Web pages as a graph, a node stands for an HTML document and an edge represents a link with an HTML label from one document (source) to another (target). In many situations it is reasonable to assume that the HTML label uniquely identifies the target document. Even if this is not literally the case, one can achieve this by including the URL (Universal Resource Locator) of the target document in the edge label. This yields a deterministic graph.

It turns out that path constraint implication has widely different complexities in these two models. In the context of $\mathcal{DM}$, the implication and finite implication problems for P_c are simple enough to be decidable in cubic-time. In contrast, in the context of $\mathcal{SM}$, these problems are hard enough to be undecidable.

Path constraint implication in the context of structured data. We investigate path constraint implication for several object-oriented models, which are similar to those studied in [3, 5, 6, 31, 58].

One important issue is the interaction between path constraints and the type system. The type system or schema definition may be viewed as imposing a type constraint on the data. For typed data, path constraint implication is considered in the context of the instances of a schema. That is, in the databases satisfying the type constraint determined by the schema. This is referred to as *path constraint implication over a schema*. Because of the impact of type constraints, in general we can no longer expect results developed for semistructured data to hold when a type is imposed on the data. Path constraint implication in the presence of types is a rich source of questions. However, although it is central to the study of path constraints on structured data, the interaction between path constraints and type constraints has received little attention.

While there has been considerable recent activity [32, 34, 43, 71] in optimizing object-oriented queries in the presence of constraints, there has, to our knowledge, been almost no work on the formulation of constraints, let alone the study of the implication problem. In [71] a rather general approach is taken: constraints are represented as boolean queries that are true, and a general framework for program optimization is used to deal with both

the optimization and the implication problem. In this setting, constraints are at least as expressive as first-order logic, and the issue of what classes of constraints have decidable implication problems is not separated from the general optimization problem.

Given the graph representation that we have adopted, we can cleanly separate typing issues from other constraints. Consider the following ODL [31] specification (loosely related to our previous XML example) which defines *Book* and *Person* classes:

interface Book
(**extent** *book*) (B1)
{ attribute String title;
relationship *set<Person> author* (B2)
inverse *Person::wrote*; (B3)
}

interface Person
(**extent** *person*) (P1)
{ attribute String name;
relationship *set<Book> wrote* (P2)
inverse *Book::author*; (P3)
}

Strike out the **extent** and **inverse** declarations at lines B1, B3, P1, P3, and change **relationship** to **attribute** on lines B2 and P2. One is now left with a standard object-oriented class/type declaration. In fact it is a declaration that can be expressed directly in a language such as C++ with type templates.

We can consider the **extent** and **inverse** declarations as added constraints:

- *Extent constraints.* For any book b, $b.author$ is a subset of the extent *person*. Similarly, for any person p, $p.wrote$ is a subset of extent *book*.
- *Inverse constraints.* For any book b and for any p in $b.author$, b is a member of

p.wrote. Similarly, for any person p and for any b in *p.wrote*, p is a member of *b.author*.

Thus, if we consider a database instance to be a graph (such as Figure 1.2 suitably modified) we can understand an ODL schema as imposing two kinds of constraints: (a) type constraints, which dictate the general structure of the graph, and (b) path constraints which dictate inclusions among certain sets of objects. We should remark that type constraints cannot be expressed as path constraints and *vice versa*.

We study the interaction between path and type constraints for two reasons. First, path constraints have proven useful for structured data. To make use of path constraints in the context of structured data, their associated implication problems should be studied in the presence of types. Second, even in the study of path constraint implication for semistructured data, there are cases in which type constraints are inevitable. For example, although the XML standard itself does not require any schema or type system, a number of proposals [16, 44, 61] have been developed that roughly correspond to data definition languages. These allow one to constrain the structure of XML data by imposing a schema or a type constraint on it. These and other proposals (e.g., [60]) also advocate the need for certain integrity constraints, which can be expressed as P_c constraints. It is likely that future XML proposals will involve both forms of constraints, and it is therefore appropriate to understand the interaction between them.

A number of results on the interaction between path and type constraints are established in this dissertation. These results show that adding structure to semistructured data may in some cases simplify reasoning about path constraints, and in other cases make it much harder.

Another issue is the impact of complex value equality on path constraint implication. In a semistructured database, it is assumed that every data entity has a unique identifier. Two data entities are equal if and only if they have identical identifiers. This is also the case in the object-oriented model considered in [3]. In some object-oriented database systems such as those studied in [5, 6, 31, 58], however, complex values with nested structures are supported. In these systems, a complex value may not have a unique identifier. Two

such data entities are equal if and only if their values are equal. This equality relation is referred to as *complex value equality*. Complex value equality can be viewed as imposing an equality constraint on the data. The presence of equality constraints raises a host of problems, and makes the analysis of path constraint implication more delicate. This issue is also investigated in this dissertation.

Axiomatization. A notion related to path constraint implication is (finite) axiomatizability. That is, the problem to determine whether there exists a (finite) set of inference rules which is sound and complete for path constraint implication and finite implication. It is desirable to develop a finite set of inference rules for a class of constraints. Inference rules can be used not only for generating symbolic proofs of implication, but also for studying the essential properties of the constraints. In general, the existence of a finite set of inference rules is a stronger property than the existence of an algorithm for testing implication. There are constraint languages for which there is no finite set of inference rules but there is an algorithm for testing their logical implication. For example, the class of unary inclusion dependencies and functional dependencies does not have a finite axiomatization for finite implication. However, there is an algorithm for testing its associated finite implication in cubic-time [36].

Several results on the finite axiomatizability of path constraints are also established in this dissertation.

1.2 Contributions

The following contributions are made.

- A path constraint language, P_c, is introduced. This language formalizes a variety of integrity constraints that commonly arise in practice and are a fundamental part of the semantics of the data. The language is of interest in connection with databases adhering to a variety of data models, ranging from sophisticated ones with richer constructs than the relational model such as object-oriented models, to semistructured models recently developed in response to the need of querying unstructured

data as found for instance on the Web. The constraints of P_c are important in query optimization. These constraints are also useful for specifying structural information about semistructured data and for ensuring information capacity preservation in database transformations.

- Abstractions of semistructured as well as structured databases are presented in terms of first-order logic. These help pave the way for applying results originating in finite model theory or various nonstandard logics to databases.

- A number of complexity results on path constraint implication are established.

  In the context of semistructured databases, two graph models are considered.

  - In $\mathcal{SM}$.

    * It is shown that despite the simple syntax of P_c, the implication problem for P_c is r.e. complete and the finite implication problem for P_c is co-r.e. complete. Indeed, two fragments of P_c are identified, and the existence of a conservative reduction from the set of all first-order sentences to each of the two fragments is established.

    * In light of these undecidability results, several fragments of P_c are identified, and the decidability of the implication and finite implication problems for each of these fragments is established. These fragments suffice to express important semantic information such as extent constraints, inverse relationships and local database constraints commonly found in object-oriented databases, and can be used for optimizing query evaluation and for specifying structural information about semistructured data. In addition, each of these fragments properly contains the class of word constraints introduced and studied in [9].

  - In $\mathcal{DM}$.

    * In contrast to the undecidability results established in $\mathcal{SM}$, it is shown that in $\mathcal{DM}$, the implication and finite implication problems for P_c coincide and

are decidable in cubic-time. In addition, there is a finite axiomatization for implication and finite implication of constraints of P_c.

  * Two generalizations of P_c are investigated. One generalization, denoted by P_c^-, allows wildcards in path expressions. The other, denoted by P_c^*, represents paths by regular expressions. It is shown that the implication finite implication problems for P_c^- are decidable, but these problems for P_c^* are undecidable.

In the context of structured databases, three object-oriented models are considered. One of them is a restricted type system, and the others are generic type systems.

- Interaction between path constraints and type constraints is investigated. It is shown that results developed for semistructured data may no longer hold in the presence of types.
  * On the one hand, there is a fragment of P_c whose associated implication and finite implication problems are decidable in PTIME in the context of $\mathcal{SM}$, but that become undecidable in the context of the two generic object-oriented models.
  * On the other hand, there is another fragment of P_c whose associated implication and finite implication problems are undecidable in $\mathcal{SM}$, but that become decidable in cubic-time in the restricted object-oriented model.
- It is shown that the implication and finite implication problems for P_c are decidable in the restricted object-oriented model, but are undecidable in the two generic models. However, the decidability of the implication and finite implication problems for word constraints is established for each of these models. In addition, in some special cases, word constraint implication is finitely axiomatizable and is decidable in PTIME.
- The impact of complex value equality is investigated. In particular, word constraint implication is studied in the context of an object-oriented model supporting complex value equality. A characterization of databases of this model

is presented in terms of equality and type constraints. It is shown that in this context, the implication and finite implication problems for word constraints remain decidable.

1.3 Related research

The idea of representing data as an edge-labeled graph and using paths to specify navigational queries dates back to the early 1980s [66, 80]. Recently, the idea has been exploited and adapted to a variety of new database applications, ranging from querying object-oriented databases (e.g., XSQL [56], OQL-doc [4]) to querying semistructured data (e.g., UnQL [20], Lorel [8], W3QS [57], MSL [69], WebSQL [65], STRUQL [41], STRUDEL [40], XML-QL [38]). Similar graph data models were also considered in [13, 37].

Deterministic graphs are also of interest in other areas. They form a special class of feature structures studied in feature logics [72]. They are also investigated in deterministic propositional dynamic logics [49, 75] and in deterministic transitive closure logics (DTC) [47, 53]. It should be mentioned that the path constraints considered here are not expressible in feature logics or in deterministic propositional dynamic logic (even with converse). Although they can indeed be expressed in DTC, in general, one cannot establish tight upper bounds for them by reducing their associated (finite) implication problems to the (finite) satisfiability problem for DTC. In fact, even two-variable DTC possesses undecidable satisfiability and finite satisfiability problems [47].

Optimization techniques for queries on semistructured data have been formally studied in [20, 73, 8, 9]. In [20], a lambda calculus for semistructured data was presented. This yields a framework for graph transformations which, in turn, allows an optimized evaluation of UnQL queries. In [73], a query decomposition method was proposed as an efficient query evaluation strategy on distributed data sources. In [8], the optimization techniques for generalized path expressions in object-oriented databases developed by [33] were considered for semistructured data. Recently, [9] investigated query optimization by using path inclusion constraints.

Structural information about semistructured data has been investigated in [19, 42, 67, 68]. In [19], a schema of a semistructured database was defined by means of graphs and simulation. Using the graph schemas, [42] provided optimization techniques for queries with regular path expressions. The problem of inferring structure of semistructured data was considered in [67, 68]. In [67], an algorithm was developed for approximately classifying objects into a type hierarchy. In [68], an approach to schema discovery by traversing navigation paths was presented.

Path constraints of P_c can be viewed as a generalization of inclusion dependencies (see, e.g., [5] for detailed discussions of inclusion dependency theory). In a relational database, inclusion dependencies compare values from different columns of one or more relations. Likewise, in a data graph, path constraints compare data entities reachable by following different navigational paths. However, the applicability of inclusion dependencies is limited to relational databases. In contrast, path constraints of P_c have proven useful in a variety of database contexts far beyond relational databases.

There has also been work on constraint languages defined in terms of paths for structured data [78, 35, 14, 54, 79, 55, 71] as well as for semistructured data [9]. A class of functional constraints, called *path functional dependencies*, was proposed in [78, 35]. The axiomatizability and decidability of its associated unrestricted implication problem were established in [78, 14, 54]. This constraint language generalizes functional dependencies in the relational data model for semantic and object-oriented data models. It differs significantly from the path constraint language P_c investigated in this dissertation, which is a generalization of (unary) inclusion dependencies in the relational model for both structured and semistructured data.

In [79, 35], a class of constraints for specifying range restrictions associated with paths, called *specialization constraints*, was proposed for object-oriented data models. The axiomatizability and decidability of its associated implication and finite implication problems were established in [55]. A specialization constraint asserts a type condition which is often specified by a schema. The central difference between specialization constraints and path constraints of P_c is that specialization constraints are type constraints for a graph repre-

senting a schema, whereas the path constraints specify inclusion relations for a graph representing data. In particular, specialization constraints must be associated with a schema, whereas the path constraints of P_c are defined for both structured and semistructured data.

Recently, a class of constraints, called EPCDs (embedded path-conjunctive dependencies), was introduced and studied for typed data in [71]. It was shown there that the implication and finite implication problems for EPCDs are decidable. The class of EPCDs is weaker than P_c in some aspects. For example, quantifier nesting is allowed in P_c but not in EPCDs. In other aspects, it is stronger than P_c. For example, non-emptiness can be expressed in EPCDs but not in P_c.

Closer to the work reported in this dissertation is the path inclusion constraint language introduced and investigated by Abiteboul and Vianu in [9]. A constraint in this language is an expression of the form $p \subseteq q$ or $p = q$, where p and q are regular expressions denoting paths in a graph representing semistructured data. In particular, if p and q are paths, i.e., sequences of labels, the constraint is called a *word constraint*. A constraint $p \subseteq q$ ($p = q$) expresses the inclusion (equality) relation between the two sets of nodes reachable along p and q. The decidability of the implication problems for the language was established in [9] for the semistructured data model $\mathcal{SM}$. In addition, [9] also showed that word constraint implication is decidable in PTIME. This constraint language differs from the constraint language P_c in expressive power. On the one hand, the constraint language of [9] allows a more general form of path expressions than P_c. On the other hand, it cannot capture inverse constraints and local database constraints, whereas these constraints can be expressed in P_c. Indeed, the language of [9] is contained in $L^2_{\infty\omega}$, the two variable fragment of the infinitary language $L_{\infty\omega}$, whereas P_c expresses constraints which are not $L^2_{\infty\omega}$ definable. Since the constraint language P_c is neither included in $L^2_{\infty\omega}$ nor categorized as a quantifier prefix fragment of first-order logic, our results concerning the implication problems for P_c are orthogonal to classical work on the decision problem for fragments of first-order logic (cf. [15]). In comparing this work to [9], it should also be noted that [9] does not consider the question of logical implication in the context of typed data.

Finally, several decidability/undecidability results of this dissertation are proved by reduction to/from well-known results established in first-order logic [15], bounded variable logics [45, 46] and language theory [51].

1.4 Organization

The remainder of this dissertation is made up of four parts. The rest of the introductory part presents some formal definitions. Parts II and III investigate path constraint implication for semistructured data and structured data, respectively. Finally, there is a concluding part.

Part I Introduction

Chapter 2 formally introduces the path constraint language, P_c. The finite axiomatizability, implication and finite implication problems for P_c are also described.

Part II Path Constraints on Semistructured Data

Chapter 3 presents two models for semistructured data: the semistructured model $\mathcal{SM}$ and the deterministic data model $\mathcal{DM}$. An abstraction of databases is introduced in terms of first-order logic for each of these models. Using this abstraction, the notion of path constraint implication is refined.

Chapters 4 and *5* study path constraint implication in the context of $\mathcal{SM}$. The results presented in these two chapters are taken from [22, 24, 25, 26].

Chapter 4 establishes the undecidability results on path constraint implication in the context of $\mathcal{SM}$. Two fragments of P_c are identified. It is shown that for each of the two fragments, the implication problem is r.e. complete and the finite implication problem is co-r.e. complete. Indeed, the existence of a conservative reduction from the set of all first-order sentences to each of the two fragments is established.

Chapter 5 identifies several fragments of the language P_c, and establishes the decidability of the implication and finite implication problems for each of these fragments in $\mathcal{SM}$. In addition, it is demonstrated that these fragments suffice to express important semantic information such as extent constraints, inverse relationships and local database constraints commonly found in object-oriented databases. It also presents a mild generalization of P_c, $P_c^{\wedge}$, and shows that the undecidability and decidability results on the fragments of P_c also hold on the analogous fragments of $P_c^{\wedge}$.

Chapter 6 investigates path constraints in the context of $\mathcal{DM}$. The decidability of the implication and finite implication problems for P_c is established. In addition, it is shown that P_c is finitely axiomatizable. A cubic-time algorithm for testing implication and finite implication of P_c constraints is presented. This chapter also introduces two generalizations of P_c, namely, P_c^- and P_c^*. The language P_c^- is defined by including wildcards in path expressions, and P_c^* is defined by representing paths as regular expressions.. It is shown that in $\mathcal{DM}$, while the implication and finite implication problems for P_c^- are decidable, these problems are undecidable for P_c^*. The results reported in this chapter were established in [30].

Part III Path Constraints on Typed Data

Chapter 7 presents three object-oriented data models which do not support complex value equality and are similar to the one studied in [3]. An abstraction of databases is introduced in terms of type constraints for each of these models. The notion of path constraint implication over a schema is also described.

Chapter 8 studies the implication and finite implication problems for P_c for each of the object-oriented models introduced in Chapter 7. It is shown that in the context of one of these models, the implication and finite implication problems for P_c are decidable in cubic-time and are finitely axiomatizable. In the two other models, these problems are undecidable. In addition, it is shown that in all three of these models, the implication and finite implication problems for the class of word constraints are decidable. Moreover, in

some special cases, word constraint implication is finitely axiomatizable and is decidable in PTIME. Some results presented in this chapter were reported in [22, 27, 28].

Chapter 9 investigates the interaction between path and type constraints. In particular, it identifies a fragment of P_c whose associated implication and finite implication problems are decidable in PTIME in $\mathcal{SM}$, but that become undecidable in some object-oriented models. In addition, it identifies another fragment of P_c whose associated implication and finite implication problems are undecidable in $\mathcal{SM}$, but that become decidable in cubic-time in an object-oriented model. The results reported in this chapter are taken from [23, 28].

Chapter 10 studies the impact of complex value equality on path constraint implication. An object-oriented model supporting complex value equality is introduced. An abstraction of the databases in this model is given in terms of both type and equality constraints. It is demonstrated that there is indeed an interaction between path and equality constraints. Finally, a small model argument is presented to establish the decidability of the implication and finite implication problems for word constraints in the presence of both type and equality constraints. The results described in this chapter were established in [29].

Part IV Conclusion

Chapter 11 summarizes the results reported in this dissertation and identifies directions for further work.

Chapter 2

Path Constraints

This chapter formally defines the path constraint language P_c. We first present the vocabulary of P_c, and then define paths and path constraints of P_c. Finally, we describe the notions of implication, finite implication and axiomatization.

2.1 Vocabulary

We assume the standard notations of variable, constant, formula, sentence, signature and structure used in first-order logic [39] and model theory [50]. The constraint language investigated in this dissertation uses predicate (relation) and constant symbols. Function symbols are not allowed. We use K and l with subscripts or superscripts for predicate symbols, and treat equality as a built-in predicate. We use x, y and z for variables, r with superscripts for constants, φ, ϕ and ψ for formulas and sentences, and G and H with subscripts for structures.

The vocabulary of the constraint language is specified by a relational signature

$$\sigma = (r,\ E),$$

where r is a constant and E is a finite set of binary relation symbols.

We specify a *σ-structure* G by giving $(|G|,\ r^G,\ E^G)$, where

- $|G|$ is a set called the *universe* (*domain*) of G. The elements of $|G|$ are called the *elements* (*nodes*) of G. We use a, b, c and o for nodes of G.
- r^G is an element of G and is called the *root node* of G.
- E^G is a set of binary relations on $|G|$, each of which is named by a relation symbol of E. For each $K \in E$, we write K^G for the relation in G named by K.

The *cardinality* (*size*) of G is defined to be the cardinality of $|G|$.

An important property of σ-structures is that there is a natural mapping between them and edge-labeled rooted directed graphs. A σ-structure G can be naturally depicted as a graph with $|G|$ as the set of vertices and E^G as the set of labeled edges. The root of the graph is r^G. In addition, for all $a, b \in |G|$ and $K \in E$, there is an edge labeled with K from a to b in the graph if and only if $(a, b) \in K^G$.

Let G and H be σ-structures. We say that G is a *substructure* of H if $|G| \subseteq |H|$, $r^G = r^H$, and for each $K \in E$, K^G is the restriction of K^H to $|G|$.

2.2 Paths

A path, i.e., a sequence of labels, can be represented as a logic formula with two free variables.

Definition 2.1: A *path* is a formula $\alpha(x, y)$ having one of the following forms:

- $x = y$, denoted by $\epsilon(x, y)$ and called the *empty path*;
- $K(x, y)$, where $K \in E$ and E is the set of binary relation symbols in signature σ; or
- $\exists z(K(x, z) \wedge \beta(z, y))$, where $K \in E$ and $\beta(z, y)$ is a path.

Here the free variables x and y denote the tail and head nodes of the path, respectively. We write $\alpha(x, y)$ as α when the parameters x and y are clear from the context. ∎

We have seen many examples of paths in Chapter 1. Among them are:

$$\exists\, z\, (Students(x,\, z) \wedge Taking(z,\, y))$$
$$\exists\, z\, (person(x,\, z) \wedge \exists\, w\, (wrote(z,\, w) \wedge \exists\, v\, (ref(w, v) \wedge author(v, y))))$$

The *concatenation* of paths $\alpha(x, z)$ and $\beta(z, y)$, denoted by $\alpha(x, z) \cdot \beta(z, y)$ or simply $\alpha \cdot \beta$,

is defined by:

$$\alpha(x,z)\cdot\beta(z,y) \;=\; \begin{cases} \beta(x,y) & \text{if } \alpha = \epsilon \\ \exists z(K(x,z) \wedge \beta(z,y)) & \text{if } \alpha = K \\ \exists u(K(x,u) \wedge (\alpha'(u,z)\cdot\beta(z,y))) & \text{if } \alpha(x,z) = \exists u(K(x,u)\wedge\alpha'(u,z)) \end{cases}$$

For example, the paths above can be written as:

$$Students \cdot Taking(x,y)$$
$$person \cdot wrote \cdot ref \cdot author(x,y)$$

We use $(\alpha)^m$ to denote the m-time concatenations of α, defined by:

$$(\alpha)^m \;=\; \begin{cases} \epsilon & \text{if } m = 0 \\ \alpha\cdot(\alpha)^{m-1} & \text{otherwise} \end{cases}$$

A path ρ is said to be a *proper prefix* of ϱ, denoted by $\rho \prec_p \varrho$, iff there exists a path λ such that $\lambda \neq \epsilon$ and $\varrho = \rho\cdot\lambda$.

A path ρ is said to be a *prefix* of ϱ, denoted by $\rho \preceq_p \varrho$, iff either $\rho \prec_p \varrho$ or $\rho = \varrho$.

Similarly, ρ is said to be a *suffix* of ϱ, denoted by $\rho \preceq_s \varrho$, iff there exists λ such that $\varrho = \lambda\cdot\rho$.

For example, the prefixes of path $person \cdot wrote \cdot ref \cdot author$ include the empty path ϵ, $person$, $person \cdot wrote$, $person \cdot wrote \cdot ref$ and $person \cdot wrote \cdot ref \cdot author$. Its suffixes include ϵ, $author$, $ref \cdot author$, $wrote \cdot ref \cdot author$ and $person \cdot wrote \cdot ref \cdot author$.

The *length* of path α, $|\alpha|$, is defined by:

$$|\alpha| \;=\; \begin{cases} 0 & \text{if } \alpha = \epsilon \\ 1 & \text{if } \alpha = K \\ 1 + |\beta| & \text{if } \alpha = K\cdot\beta \end{cases}$$

For example, $|Students \cdot Taking| = 2$ and $|person \cdot wrote \cdot ref \cdot author| = 4$.

In particular, a path of either the form $\alpha(r,x)$ or $\alpha(x,r)$, where r is the constant in σ, can be expressed as a formula with at most two distinct variables. For example, the path

$Students \cdot Taking \cdot Enrolled(r, x)$ can be expressed as:

$$\exists\, y\, (Enrolled(y, x) \wedge \exists\, x\, (Taking(x, y) \wedge Students(r, x)))$$

This formula uses only two distinct variables.

2.3 Path constraint language P_c

The path constraint language P_c is formalized as follows.

Definition 2.2: A *path constraint* φ is an expression of either the *forward* form

$$\forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(x,\, y))),$$

or the *backward* form

$$\forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x))),$$

where α, β, γ are paths, called the *prefix*, *left tail* and *right tail* of φ, and denoted by $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$, respectively.

A path constraint is called a *forward constraint* if it is of the forward form, and is called a *backward constraint* if it is of the backward form.

The set of all path constraints is denoted by P_c. ■

For example, all the path constraints presented in Chapter 1 (except (†)) are constraints of P_c.

As demonstrated in Chapter 1, path constraints of P_c are capable of expressing extent, inverse and local database constraints.

Next, we identify several special subclasses of P_c.

We call a path constraint φ of P_c a *simple path constraint* if $pf(\varphi) = \epsilon$. That is, φ is of either the form

$$\forall\, y\, (\beta(r, y) \rightarrow \gamma(r, y)),$$

or the form

$$\forall\, y\, (\beta(r, y) \rightarrow \gamma(y, r)).$$

The set of all simple path constraints is denoted by P_s.

A proper subclass of simple path constraints, called *word constraints*, was introduced and investigated in [9].

Definition 2.3: A *word constraint* φ is an expression of the form

$$\forall\, y\, (\beta(r, y) \rightarrow \gamma(r, y)),$$

where β and γ are paths, denoted by $lt(\varphi)$ and $rt(\varphi)$, respectively.

The set of all word constraints is denoted by P_w. ∎

In other words, a word constraint is a forward path constraint of P_c with its prefix being the empty path ϵ.

As demonstrated in Chapter 1, extent constraints can be expressed as word constraints.

2.4 Implication and finite implication

We borrow the standard notions of model and implication from first-order logic [39].

Let G be a structure and φ a sentence. We use $G \models \varphi$ to denote that G *satisfies* φ (i.e., G *is a model of* φ). Let Σ be a set of sentences. We use $G \models \Sigma$ to denote that G *satisfies* Σ (i.e., G *is a model of* Σ). That is, for every $\phi \in \Sigma$, $G \models \phi$.

Let $\Sigma \cup \{\varphi\}$ be a finite set of sentences. We use $\Sigma \models \varphi$ to denote that Σ *implies* φ. That is, for every structure G, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_f \varphi$ to denote that Σ *finitely implies* φ. That is, for every finite structure G, if $G \models \Sigma$, then $G \models \varphi$.

Let C be a class of sentences. The *implication problem for* C is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of C, whether $\Sigma \models \varphi$. Similarly, the *finite implication problem for* C is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of C, whether $\Sigma \models_f \varphi$.

The central technical problems investigated in this dissertation are the implication and finite implication problems for P_c and fragments thereat.

The notion of implication and finite implication for P_c shall be refined in Parts II and III.

2.5 Axiomatization

We shall also use the notion of axiomatization (see [5, 36, 48] for detailed discussions of this notion).

Let C be a class of sentences. An *axiomatization* of C is a set of inference rules $\mathcal{I}$ that is used to syntactically derive a sentence φ of C from a finite subset Σ of C. We write $\Sigma \vdash_{\mathcal{I}} \varphi$ if φ is derivable (provable) from Σ using $\mathcal{I}$. The sequence of sentences occurring in the derivation is called an *$\mathcal{I}$-proof* of φ from Σ. An axiomatization $\mathcal{I}$ (or a set of inference rules $\mathcal{I}$) is *sound for (finite) implication of C* if for each finite subset $\Sigma \cup \{\varphi\}$ of C, when $\Sigma \vdash_{\mathcal{I}} \varphi$, then $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$). An axiomatization $\mathcal{I}$ is *complete for (finite) implication of C* if for each finite subset $\Sigma \cup \{\varphi\}$ of C, when $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$), then $\Sigma \vdash_{\mathcal{I}} \varphi$.

An axiomatization is said to be *finite* if it consists of finitely many inference rules.

A class of sentences C is said to be *axiomatizable for (finite) implication* if there is an axiomatization of C that is both sound and complete for (finite) implication of C. The class C is said to be *axiomatizable* if there is an axiomatization of C that is sound and complete for both implication and finite implication of C. Similarly, C is said to be *finitely axiomatizable* if there is a finite axiomatization of C that is sound and complete for both implication and finite implication of C.

We shall use the following well-known results (see, e.g., [5]). Let C be a class of sentences and $\mathcal{I}$ a finite set of inference rules.

- If $\mathcal{I}$ is sound and complete for finite implication of C, then the finite implication problem for C is decidable.
- If $\mathcal{I}$ is sound and complete for finite implication of C, and moreover, $\mathcal{I}$ is sound for implication of C, then both the implication problem and the finite implication

problem for C are decidable.

Several results on the finite axiomatizability of P_c and P_w in certain contexts are established in this dissertation.

Part II

Path Constraints on Semistructured Data

The goal of Part II is to settle the question of path constraint implication in the context of semistructured databases. The implication and finite implication problems for path constraints are the most important theoretical issues in connection with path constraints. In Part II, we limit the discussions to semistructured data, by which we mean data whose structure is not constrained by a schema. Recently, semistructured data has arisen considerable interest. The quest for semistructured data naturally arises from a variety of applications, including data browsing, data exchange and integration, and querying unstructured data as found for instance on the World-Wide Web and in biological databases.

We consider two models for semistructured data. One represents data as an edge-labeled, rooted, directed graph. This model is referred to as the *semistructured data model* ($\mathcal{SM}$). The other is a variant of $\mathcal{SM}$, called the *deterministic data model* ($\mathcal{DM}$). In $\mathcal{DM}$, data is represented as a graph in $\mathcal{SM}$ with deterministic edge relations. That is, the edges emanating from any node in the graph have distinct labels. These two models are formally presented in Chapter 3.

Chapters 4 and 5 investigate path constraint implication for $\mathcal{SM}$. More specifically, Chapter 4 shows that, despite the simple syntax of P_c, the implication problem for P_c is r.e. complete, and the finite implication problem for P_c is co-r.e. complete. These results are rather surprising since P_c is a mild generalization of P_w, which possesses decidable implication and finite implication problems in $\mathcal{SM}$ [9].

These undecidability results motivate our search for decidable fragments of P_c which retain sufficient expressive power to make them of interest from a database perspective. Chapter 5 identifies several fragments of P_c, and establishes the decidability of the implication

and finite implication problems associated with each of these fragments in $\mathcal{SM}$. It also demonstrates that these fragments suffice to express many important integrity constraints, including extent, inverse and local databases constraints discussed in Chapter 1. In addition, in Chapter 5 we investigate the class of conjunctive path constraints, $P_c^{\wedge}$, which is a generalization of P_c. We show that the undecidability and decidability results on the fragments of P_c investigated here also hold on the analogous fragments of $P_c^{\wedge}$.

Chapter 6 studies path constraint implication for $\mathcal{DM}$. In contrast to the undecidability results established in Chapter 4, it is shown that in $\mathcal{DM}$, the implication and finite implication problems for P_c are not only decidable in cubic-time, but also finitely axiomatizable. This demonstrates that the determinism condition of $\mathcal{DM}$ simplifies the analysis of path constraint implication. This chapter also investigates two generalization of P_c, namely, P_c^- and P_c^*. The language P_c^- allows wildcards in path expressions, and P_c^* is defined in terms of regular expressions. It is shown that for P_c^-, the implication and finite implication problems are decidable. However, these problems for P_c^* remain undecidable. This undecidability result shows that the determinism condition of $\mathcal{DM}$ does not trivialize the problem of path constraint implication.

Chapter 3

Semistructured Data Models

This chapter presents two models for semistructured data. One is the well-known graph model, often referred to as the semistructured data model ($\mathcal{SM}$) in the literature [2, 18]. Interest in the other model, called the deterministic data model ($\mathcal{DM}$), is rather recent [21]. We give an abstraction of databases in terms of first-order logic for each of these models, and refine the notion of path constraint implication in the context of $\mathcal{SM}$ and $\mathcal{DM}$. We also introduce two extensions of P_c, denoted by P_c^- and P_c^*, respectively, which will be investigated in Chapter 6 for $\mathcal{DM}$.

3.1 The semistructured data model $\mathcal{SM}$

Semistructured data is usually modeled as an edge-labeled, rooted, directed graph, e.g., in UnQL [20] and in OEM [8, 68, 70]. Along the same lines, here we use an abstraction of semistructured databases as (finite) first-order logic structures of signature

$$\sigma = (r,\ E)$$

described in Chapter 2.

In a σ-structure G, the constant r^G indicates an entry point into the database represented by G, and the universe $|G|$ represents the set of data entities of the database. Structure G can be depicted as a rooted edge-labeled directed graph with $|G|$ as the set of vertices, E^G the set of labeled edges and r^G the root.

We call this model the *semistructured data model*, abbreviated to $\mathcal{SM}$.

We refine the notion of path constraint implication in $\mathcal{SM}$ as follows. Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c. We use $\Sigma \models_s \varphi$ to denote that Σ *implies* φ *in* $\mathcal{SM}$. That is, for every

σ-structure G, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_{(s,f)} \varphi$ to denote that Σ *finitely implies* φ *in* $\mathcal{SM}$. That is, for every finite σ-structure G, if $G \models \Sigma$, then $G \models \varphi$.

In $\mathcal{SM}$, the *implication problem* for P_c is the problem of determining, given any finite set $\Sigma \cup \{\varphi\}$ of constraints in P_c, whether all σ-structures that satisfy Σ are also models of φ ($\Sigma \models_s \varphi$). The *finite implication problem* for P_c is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of P_c, whether all finite σ-structures that satisfy Σ are also models of φ ($\Sigma \models_{(s,f)} \varphi$).

We write $\Sigma \models_s \varphi$ ($\Sigma \models_{(s,f)} \varphi$) simply as $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) if the context $\mathcal{SM}$ is understood.

In the context of $\mathcal{SM}$, the implication and finite implication problems for path constraints are investigated in Chapters 4 and 5.

3.2 The deterministic data model $\mathcal{DM}$

For data found in many applications, the graph is *deterministic*, i.e., the edges emanating from each node in the graph have distinct labels. For example, when modeling Web pages as a graph, a node stands for an HTML document and an edge represents a link with an HTML label from one document (source) to another (target). It is reasonable to assume that the HTML label uniquely identifies the target document. Even if this is not literally the case, one can achieve this by including the URL (Universal Resource Locator) of the target document in the edge label. This yields a deterministic graph. As another example, consider ACeDB [74], which is a database management system popular with biologists. A graph representing an ACeDB database is also deterministic. In general, any database with "exportable" data identities can be modeled as a deterministic graph by including the identities in the edge labels. Here by exportable identities we mean directly observable identities such as keys. Some relational and object-oriented database management systems support exportable identities. In particular, in the OEM model (see, e.g., [8]), there are exportable object identities. To capture this, we consider a data model for semistructured data which is a variant of SM, referred to as *the deterministic data model* ($\mathcal{DM}$). In $\mathcal{DM}$, data is represented as a deterministic, rooted, edge-labeled, directed graph. That is, for

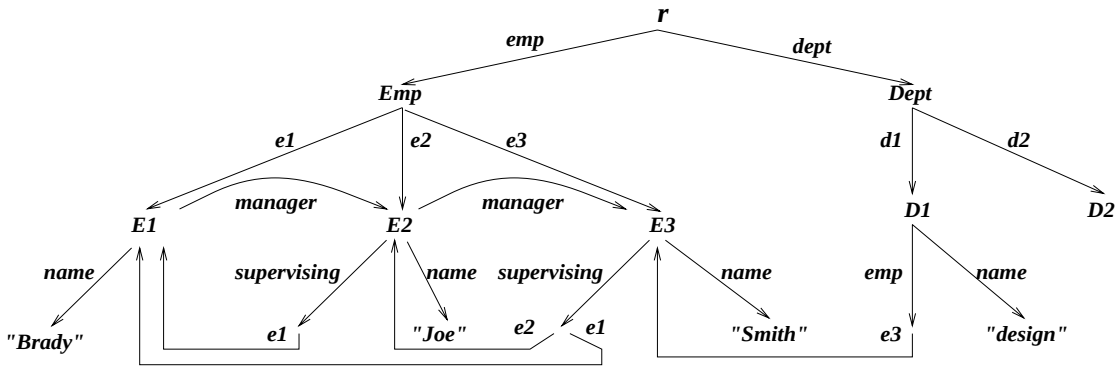

Figure 3.1: An example semistructured database in $\mathcal{DM}$

each node a and label K, there exists at most one edge labeled K going out of a. An important feature of $\mathcal{DM}$ is that in this model, each component of a database is uniquely identified by a path.

Along the same lines of the abstraction of databases in SM, we represent a database in $\mathcal{DM}$ as a (finite) σ-structure satisfying the *determinism condition*:

$$\bigwedge_{K \in E} \forall\, x\, y\, z\, (K(x,\, y) \wedge K(x,\, z) \rightarrow y = z).$$

Such structures are called *deterministic structures*.

In the context of $\mathcal{DM}$, we also study two extensions of P_c, denoted by P_c^- and P_c^*, respectively. Below we first use an example to illustrate how P_c^- and P_c^* constraints can be used in query optimization in the context of $\mathcal{DM}$, and then formally define P_c^- and P_c^*. Finally, we refine the notion of path constraint implication in the context of $\mathcal{DM}$.

3.2.1 An example.

To demonstrate applications of path constraints in the context of $\mathcal{DM}$, let us consider Figure 3.1, which collects information on employees and departments. It is an example of semistructured data represented in the deterministic data model. In Figure 3.1, there are two edges emanating from the root node r, which are labeled `emp` and `dept` and connected to nodes `Emp` and `Dept`, respectively. Edges emanating from `Emp` are labeled with employee ID's and connected to vertices representing employees. An employee node may have three

edges emanating from it: an edge labeled `manager` and connected to his/her manager, an edge labeled `supervising` that connects to a node from which there are outgoing edges connected to employees under his/her supervision, and an edge labeled `name`. Similarly, there are vertices representing departments that may have edges connected to employees. Observe that Figure 3.1 is deterministic.

Path constraints. Typical path constraints on Figure 3.1 include:

$$\forall x\,(emp \cdot \_ \cdot manager(r,\, x) \rightarrow emp \cdot \_\,(r,\, x)) \qquad (\phi_1)$$

$$\forall x\,(emp \cdot \_ \cdot supervising \cdot \_\,(r,\, x) \rightarrow emp \cdot \_\,(r,\, x)) \qquad (\phi_2)$$

$$\forall x\,(emp \cdot \_\,(r,\, x) \rightarrow \forall y\,(manager(x,\, y) \rightarrow supervising \cdot \_\,(y,\, x))) \qquad (\phi_3)$$

Here "_" is a "wildcard" symbol, which matches any edge label. Path formulas can be naturally generalized to include wildcards. The path constraints above describe inclusion relations. More specifically, ϕ_1 states that if a node is reached from the root r by following $\texttt{emp}\cdot\_\cdot\texttt{manager}$, then it is also reachable from r by following $\texttt{emp}\cdot\_$. It asserts that the manager of any employee is also an employee that occurs in the database. Similarly, ϕ_2 states that if a node is reached from r by following $\texttt{emp}\cdot\_\cdot\texttt{supervising}\cdot\_$, then it is also reachable from r by following $\texttt{emp}\cdot\_$. constraint ϕ_3 states that for any employee x and for any y, if x is connected to y by a `manager` edge, then x is reachable from y by following $\texttt{supervising}\cdot\_$. These are constraints of P_c^-, one of generalizations of P_c that will be studied in Chapter 6.

We generalize P_c^- by representing paths as regular expressions. This generalization is denoted by P_c^*. For example, the following are constraints of P_c^*:

$$\forall x\,(emp \cdot \_\,(r,\, x) \rightarrow \forall y\,(manager \cdot manager^*(x,\, y) \rightarrow supervising \cdot \_\,(y,\, x))) \qquad (\psi_1)$$

$$\forall x\,(emp \cdot \_\,(r,\, x) \rightarrow \forall y\,(supervising \cdot \_\,(x,\, y) \rightarrow manager \cdot manager^*(y,\, x))) \qquad (\psi_2)$$

Here $*$ is the Kleene closure. These constraints describe an inverse relationship between $\texttt{manager}\cdot\texttt{manager}^*$ and $\texttt{supervising}\cdot\_$. More specifically, ψ_1 asserts that for any employee x and for any y, if y is reachable from x by following one or more `manager` edges,

then x is reachable from y by following path `supervising`$\cdot$`_`. Similarly, ψ_2 asserts that if y is reachable from x by following `supervising`$\cdot$`_`, then x is reachable from y by following one or more `manager` edges. The constraint language P_c^* will also be studied in Chapter 6.

As opposed to P_c^* constraints, path constraints of P_c contain neither wildcards nor the Kleene star. In the deterministic data model, P_c constraints express path equalities. For example, the following can be described by P_c constraints:

$$emp \cdot e1 \cdot manager = emp \cdot e2 \qquad (\varphi_1)$$
$$dept \cdot d1 \cdot emp \cdot e1 = emp \cdot e1 \qquad (\varphi_2)$$

These can also be expressed as word constraints. Observe that the paths in the P_c constraints above do not contain wildcards and the Kleene closure.

Semantic specification with path constraints. The path constraints above describe certain typing information about the data. For example, abusing object-oriented database terms, ϕ_1 asserts that a manager of an employee has an "`employee` type", and in addition, is in the "extent" of "class" `employee`. By using ϕ_1, it can be shown that for any employee x and any y, if y is reachable from x by following zero or more `manager` edges, then y also has an "`employee` type" and is in the "extent" of `employee`. A preliminary type system was proposed in [21] for the deterministic data model, in which the types of paths are defined by means of path constraints. This is a step in unifying the (programming language) notion of a type with the (database) notion of a schema.

Query optimization with path constraints. To illustrate how path constraints can be used in query optimization, consider again the database represented in Figure 3.1. Suppose, for example, we want to find the name of the employee with ID $e1$ in department $d1$. One may write the query as Q_1 (in Lorel syntax [8]):

Q_1:
```
select X.name
from   r.dept.d1.emp.e1 X
```

Given path constraint φ_2, the query Q_1 can be rewritten as Q_1':

Q_1':
```
select X.name
from   r.emp.e1 X
```

One can easily verify that Q_1 and Q_1' are equivalent.

As another example, suppose we want to find the names of the employees connected to Smith by one or more `manager` edges. Without path constraints, one would write the query as Q_2 (in Lorel syntax):

Q_2:
```
select X.name
from   r.emp.% X, X(.manager)+ Y
where  Y.name = "Smith"
```

In Lorel, `%` denotes wildcard and `(.manager)+` means one or more occurrences of `.manager`. Given constraints ψ_1, ψ_2, ϕ_1 and ϕ_2, we can rewrite Q_2 as Q_2', which finds the names of the employees under the supervision of Smith:

Q_2':
```
select X.name
from   r.emp.% Y, Y.supervising.% X
where  Y.name = "Smith"
```

It can be verified that given those path constraints, Q_2 and Q_2' are equivalent. In addition, Q_2' is more efficient than Q_2 because it does not require the traversal of sequences of `manager` edges. It should be mentioned that to show that Q_2 and Q_2' are equivalent, we need to verify that certain constraints necessarily hold given that ψ_1, ψ_2, ϕ_1 and ϕ_2 hold. That is, they are implied by ψ_1, ψ_2, ϕ_1 and ϕ_2. In particular, we need to show that ψ_3 below is implied by ψ_1, ψ_2, ϕ_1 and ϕ_2:

$$\forall x\,(emp \cdot \_ \cdot manager^*(r,\, x) \rightarrow emp \cdot \_\,(r,\, x)) \qquad (\psi_3)$$

3.2.2 Path constraint languages P_c^- and P_c^*

We next formally define P_c^- and P_c^*.

Path constraint language P_c^*. To define P_c^*, we first generalize the syntax of path expressions.

We represent path expressions as regular expressions, defined by the syntax:

$$e \ ::= \ \epsilon \mid K \mid e \cdot e \mid e + e \mid e^*$$

Here ϵ is the empty path, $K \in E$ (E is the set of binary relation symbols in σ), $\cdot$, $+$ and $*$ represent concatenation, union and the Kleene closure, respectively.

Let p be a regular expression and ρ be a path. We use $\rho \in p$ to denote that ρ is in the regular language generated by p.

We also treat a regular expression p as a logic formula $p(x, y)$, where x and y are free variables. A deterministic structure G *satisfies* $p(x, y)$, denoted by $G \models p(x, y)$, if there exist path $\rho \in p$ and nodes $a, b \in |G|$ such that $G \models \rho(a, b)$.

Recall that the wildcard symbol "_" matches any edge label. We can express "_" as a regular expression. More specifically, let E, the finite set of binary relation symbols in signature σ, be enumerated as K_1, K_2, ..., K_n. Then "_" can be defined as the regular expression:

$$K_1 + K_2 + ... + K_n.$$

In Section 3.2.1, we have seen the following path expressions that can be represented as regular expressions:

$$manager \cdot manager^*$$
$$emp \cdot \_ \cdot manager^*$$

Using regular expressions, we define P_c^* as follows.

Definition 3.1: A constraint ψ of P_c^* is an expression of either the *forward form*:

$$\forall x \, (p(r, x) \rightarrow \forall y \, (q(x, y) \rightarrow s(x, y))),$$

or the *backward form:*

$$\forall x \, (p(r, x) \rightarrow \forall y \, (q(x, y) \rightarrow s(y, x))),$$

where p, q and s are regular expressions, denoted by $pf(\psi)$, $lt(\psi)$ and $rt(\psi)$, respectively.

■

For example, all the path constraint given in Section 3.2.1 are (or can be expressed as) P_c^* constraints.

A deterministic structure G *satisfies* a constraint ϕ of P_c^*, denoted by $G \models \phi$, if the following condition is satisfied:

- when ϕ is a forward constraint: for all $a, b \in |G|$, if there exist paths $\alpha \in p$ and $\beta \in q$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$, then there exists a path $\gamma \in s$ such that $G \models \gamma(a, b)$;
- when ϕ is a backward constraint: for all $a, b \in |G|$, if there exist paths $\alpha \in p$ and $\beta \in q$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$, then there exists $\gamma \in s$ such that $G \models \gamma(b, a)$.

Path constraint language P_c^-. We next present P_c^-. Path constrains of P_c^- are defined in terms of path expressions represented by a restricted form of regular expressions:

$$w ::= \epsilon \mid K \mid w \cdot w \mid w + w$$

That is, we define path expressions to be regular expressions which do not contain the Kleene closure. Let us refer to such expressions as $*$-*free regular expressions*.

The following should be noted about $*$-free regular expressions.

- The regular language generated by a $*$-free regular expression is finite.
- The wildcard symbol "_" can be expressed as, in fact, a $*$-free regular expression.

For example, we have seen in Section 3.2.1 the following path expressions that can be represented as $*$-free regular expressions:

$$emp \cdot \_ \cdot manager$$
$$emp \cdot \_ \cdot supervising \cdot \_$$

Using $*$-free regular expressions, we define P_c^- as follows.

Definition 3.2: The path constraint language P_c^- is defined to be

$$P_c^- = \{\phi \mid \phi \in P_c^*,\ pf(\phi),\ lt(\phi) \text{ and } rt(\phi) \text{ are } *\text{-free regular expressions}\},$$

where $pf(\phi)$, $lt(\phi)$ and $rt(\phi)$ are described in Definition 3.1. ∎

For example, path constraints ϕ_1, ϕ_2 and ϕ_3 given in Section 3.2.1 are P_c^- constraints, but ψ_1, ψ_2 and ψ_3 are not in P_c^-.

Path constraint languages P_c and P_w. As opposed to P_c^- and P_c^* constraints, P_c and P_w constraints are defined in terms of paths, i.e., sequences of edge labels, which are syntactically defined by:

$$\rho \ ::= \ \epsilon \mid K \mid K \cdot \rho$$

In other words, they are defined in terms of regular expressions containing neither the wildcard symbol nor the Kleene closure. The languages P_c and P_w are restrictions of P_c^-. More specifically,

$$\begin{aligned} P_c &= \{\varphi \mid \varphi \in P_c^-,\ pf(\varphi),\ lt(\varphi) \text{ and } rt(\varphi) \text{ are paths}\}, \\ P_w &= \{\varphi \mid \varphi \in P_c,\ \varphi \text{ is a forward constraint, } pf(\varphi) = \epsilon\}. \end{aligned}$$

For example, constraints φ_1 and φ_2 given in Section 3.2.1 can be described by P_c constraints (in fact, P_w constraints):

$$\begin{aligned} &\forall\, x\, (emp \cdot e1 \cdot manager(r,\ x) \rightarrow emp \cdot e2(r,\ x)) \\ &\forall\, x\, (emp \cdot e2(r,\ x) \rightarrow emp \cdot e1 \cdot manager(r,\ x)) \\ &\forall\, x\, (dept \cdot d1 \cdot emp \cdot e1(r,\ x) \rightarrow emp \cdot e1(r,\ x)) \\ &\forall\, x\, (emp \cdot e1(r,\ x) \rightarrow dept \cdot d1 \cdot emp \cdot e1(r,\ x)) \end{aligned}$$

Note that ψ_1, ψ_2, ψ_3, ϕ_1, ϕ_2 and ϕ_3 are not P_c constraints.

Summary. Comparing the expressive powers of these constraint languages, we have

$$P_w \ \subset \ P_c \ \subset \ P_c^- \ \subset \ P_c^*$$

It should be noted that while $*$-free regular expressions, P_w, P_c and P_c^- are definable in first-order logic, regular expressions and P_c^* are not.

3.2.3 Path constraint implication in $\mathcal{DM}$.

Next, we refine the notion of path constraint implication in $\mathcal{DM}$.

Let $C \in \{P_w, P_c, P_c^-, P_c^*\}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of C. We use $\Sigma \models_d \varphi$ to denote that Σ *implies* φ *in* $\mathcal{DM}$. That is, for every deterministic structure G, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_{(d,f)} \varphi$ to denote that Σ *finitely implies* φ *in* $\mathcal{DM}$. That is, for every finite deterministic structure G, if $G \models \Sigma$, then $G \models \varphi$.

In $\mathcal{DM}$, the *implication problem* for C is the problem of determining, given any finite set $\Sigma \cup \{\varphi\}$ of constraints in C, whether all the deterministic structures that satisfy Σ are also models of φ ($\Sigma \models_d \varphi$). The *finite implication problem* for C is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of C, whether all the finite deterministic structures that satisfy Σ are also models of φ ($\Sigma \models_{(d,f)} \varphi$).

An axiomatization $\mathcal{I}$ is said to be *sound for (finite) implication of* C *in* $\mathcal{DM}$ if for each finite subset $\Sigma \cup \{\varphi\}$ of C, when $\Sigma \vdash_{\mathcal{I}} \varphi$, then $\Sigma \models_d \varphi$ ($\Sigma \models_{(d,f)} \varphi$). An axiomatization $\mathcal{I}$ is *complete for (finite) implication of* C *in* $\mathcal{DM}$ if for each finite subset $\Sigma \cup \{\varphi\}$ of C, when $\Sigma \models_d \varphi$ ($\Sigma \models_{(d,f)} \varphi$), then $\Sigma \vdash_{\mathcal{I}} \varphi$.

We write $\Sigma \models_d \varphi$ ($\Sigma \models_{(d,f)} \varphi$) simply as $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) if the context $\mathcal{DM}$ is understood.

For example, let $\Sigma = \{\psi_1, \psi_2, \phi_1, \phi_2\}$, where ψ_1, ψ_2, ϕ_1 and ϕ_2 are given in Section 3.2.1. Then the question whether $\Sigma \models \psi_3$ ($\Sigma \models_f \psi_3$) is an instance of the (finite) implication problem for P_c^*. In Section 3.2.1, this implication is used in the proof of the equivalence of the queries Q_2 and Q_2'.

In the context of $\mathcal{DM}$, the implication and axiomatization of path constraints are investigated in Chapter 6.

Chapter 4

Undecidable Implication Problems in $\mathcal{SM}$

This chapter investigates the implication and finite implication problems for P_c in the context of $\mathcal{SM}$. We show that, despite the simple syntax of P_c, both the implication and finite implication problems for P_c are undecidable. We begin by presenting two sublanguages of P_c, P_f and P_+, in Section 4.1. We then establish the undecidability of the implication and finite implication problems for P_+ and P_f in Sections 4.2 and 4.3, respectively. More specifically, we show that for each of these sublanguages, the implication problem is r.e. complete and the finite implication problem is co-r.e. complete.

4.1 Undecidable subclasses of P_c

In this section, we first identify two sublanguages of P_c, for which we shall show that the implication and finite implication problems are undecidable. We then review the definitions of two-register machines and conservative reductions, which are used when establishing the undecidability results.

As observed by [9], every word constraint (in fact, every simple path constraint) can be expressed by a sentence in two-variable first-order logic (FO^2), the fragment of first-order logic consisting of all relational sentences with at most two distinct variables (see [15, 45] for in-depth presentations of FO^2). Recently, [45] has shown that the satisfiability problem for FO^2 is NEXPTIME-complete by establishing that any satisfiable FO^2 sentence has a model of size exponential in the length of the sentence. The decidability of the implication and finite implication problems for word constraints (and for simple constraints) in $\mathcal{SM}$ follows immediately. In fact, Abiteboul and Vianu have directly established (without reference to the embedding into FO^2) that the implication problems for word constraints are in PTIME [9].

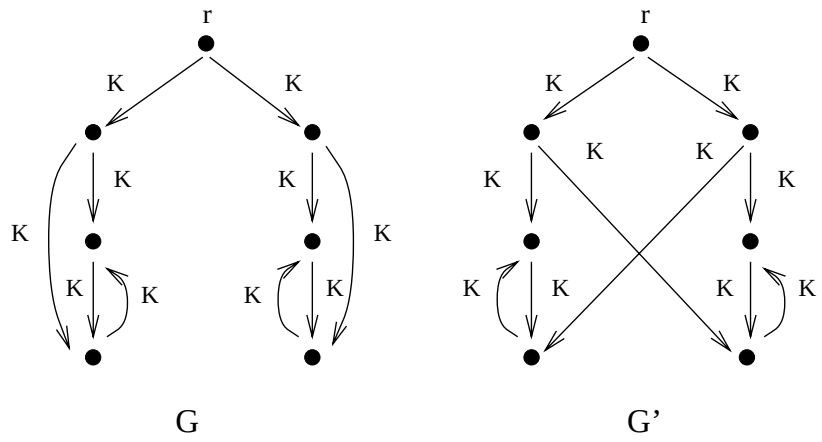

Figure 4.1: Structures distinguishable by P_c

A class of more general path constraints, denoted by P_w^*, was also studied in [9]. A constraint of P_w^* has either the form $p \subseteq q$ or $p = q$, where p and q are regular expressions representing paths. Let p be a regular expression and ρ be a path. We use $\rho \in p$ to denote that ρ is in the regular language generated by p. A structure G satisfies a P_w^* constraint $p \subseteq q$ iff for any $a \in |G|$, if there exists paths $\alpha \in p$ such that $G \models \alpha(r^G, a)$, then there exists a path $\beta \in q$ such that $G \models \beta(r^G, a)$. A structure G satisfies $p = q$ iff G satisfies both $p \subseteq q$ and $q \subseteq p$. It is easy to verify that P_w^* is contained in $L^2_{\infty\omega}$, the two variable fragment of the infinitary language $L_{\infty\omega}$. It was shown in [9] that the implication and finite implication problems for P_w^* coincide and are decidable in EXPSPACE in the context of $\mathcal{SM}$.

In contrast to word constraints, many path constraints in P_c are not expressible in FO^2 or even in $L^2_{\infty\omega}$.

Example 4.1: Consider the structures G and G' given in Figure 4.1. It is easy to verify, using the 2-pebble Ehrenfeucht-Fraïssé style game [12, 52], that G and G' are equivalent in FO^2 and $L^2_{\infty\omega}$. However, G and G' are distinguished by the P_c constraint

$$\varphi = \forall x\,(K(r,x) \rightarrow \forall y\,(K(x,y) \rightarrow \exists z(K(x,z) \wedge K(z,y)))),$$

because $G \models \varphi$ but $G' \not\models \varphi$. This shows that φ is expressible in neither FO^2 nor $L^2_{\infty\omega}$. ■

In addition, both the implication and finite implication problems for P_c are undecidable.

Theorem 4.1: In the context of $\mathcal{SM}$, the implication problem for P_c is r.e. complete, and the finite implication problem for P_c is co-r.e. complete. ∎

4.1.1 Sublanguages P_+ and P_f

In fact, the undecidability results given above also hold for two proper subclasses of P_c, which are defined below.

One of the subclasses, P_f, is the set of all the constraints of P_c having the forward form. The other, P_+, is the set

$$\{\varphi \mid \varphi \in P_c,\ lt(\varphi) \neq \epsilon,\ rt(\varphi) \neq \epsilon\}.$$

Here the notations $lt(\varphi)$ and $rt(\varphi)$ are described in Definition 2.2. Note that P_+ is the largest subset of P_c without equality.

For P_+ and P_f we have the following theorems, from which Theorem 4.1 follows immediately.

Theorem 4.2: In the context of $\mathcal{SM}$, the implication problem for P_+ is r.e. complete, and the finite implication problem for P_+ is co-r.e. complete. ∎

Theorem 4.3: In the context of $\mathcal{SM}$, the implication problem for P_f is r.e. complete, and the finite implication problem for P_f is co-r.e. complete. ∎

We prove Theorems 4.2 and 4.3 in Sections 4.2 and 4.3, respectively. The idea of the proofs is to establish a reduction from the halting problem for two-register machines [15]. In other words, we prove the existence of a conservative reduction from the set of all first-order sentences to each of the two sublanguages.

Before we present the detailed proofs, we first recall the definitions of two-register machines and conservative reduction classes given in [1, 15].

4.1.2 Two-register machines

A two-register machine (2-RM) M consists of two registers $register_1, register_2$, and is programmed by a numbered sequence $I_0, I_1, ..., I_l$ of instructions. Each register contains a natural number.

An *instantaneous description* (ID) of M is (i, m, n), where $i \in [0, l]$, m and n are natural numbers. It indicates that M is ready to execute instruction I_i (or at "state i") with $register_1$ and $register_2$ containing m and n, respectively.

An instruction of M is either an *addition* or a *subtraction*, which defines a relation $\rightarrow_M$ between IDs, described as follows:

- *addition*: (i, rg, j), where rg is either $register_1$ or $register_2$, $0 \leq i, j \leq l$. The semantics of the addition instruction is: at state i, M adds 1 to the content of rg, and then goes to state j. Accordingly:

$$(i, m, n) \rightarrow_M \begin{cases} (j, m+1, n) & \text{if } rg = register_1 \\ (j, m, n+1) & \text{otherwise} \end{cases}$$

- *subtraction*: (i, rg, j, k), where rg is either $register_1$ or $register_2$, $0 \leq i, j, k \leq l$. The semantics of the subtraction instruction is: at state i, M tests whether the content of rg is 0, and if it is, then goes to state j; otherwise M subtracts 1 from the content of rg and goes to the state k. Accordingly:

$$(i, m, n) \rightarrow_M \begin{cases} (j, 0, n) & \text{if } rg = register_1 \text{ and } m = 0 \\ (k, m-1, n) & \text{if } rg = register_1 \text{ and } m \neq 0 \\ (j, m, 0) & \text{if } rg = register_2 \text{ and } n = 0 \\ (k, m, n-1) & \text{if } rg = register_2 \text{ and } n \neq 0 \end{cases}$$

The relation $\rightarrow_M$ can be understood as a set of rewrite rules for IDs.

We use $\Rightarrow_M$ to denote the reflexive and transitive closure of $\rightarrow_M$. The relation of *M-reachability* $C \Rightarrow_M D$ holds just in case M, started from ID C, reaches ID D by application of zero or more $\rightarrow_M$ rewrite rules.

To describe the halting problem for 2-RMs, we borrow the following notations from [1, 15].

Definition 4.1 [1, 15]: Let X be a class of sentences. We write

- $N(X)$ for the set of all *unsatisfiable* sentences in X; that is,

$$N(X) = \{\psi \mid \psi \in X,\ \psi \text{ does not have a model}\};$$

- $F(X)$ for the set of all *finitely satisfiable* sentences in X; that is,

$$F(X) = \{\psi \mid \psi \in X,\ \psi \text{ has a finite model}\}.$$

We write FO for the set of all first-order sentences. ∎

Recall the following well-known result [77]:

> There is an effective partial procedure by which, given a sentence in FO, we can test whether it has no model, a finite model, or only infinite models. The procedure terminates in the first two cases, but does not terminate in the last case.

We fix M_L to be a two-register machine with the following behavior (the existence of such a machine follows from the result just quoted. See [1, 15] for further discussion). The two-register machine M_L has two halting states: $(1, 0, 0)$ and $(2, 0, 0)$. For each $\psi \in FO$, let $m(\psi)$ be an appropriate encoding of ψ (a natural number) and $C(\psi)$ be the ID $(0, m(\psi), 0)$ of M_L. Started from $C(\psi)$,

- M_L halts at (1, 0, 0) iff ψ is not satisfiable; and
- M_L halts at (2, 0, 0) iff ψ has a finite model.

In other words, M_L has the following property. For $i = 1, 2$, let

$$H_{M_L,i} = \{\psi \mid \psi \in FO,\ C(\psi) \Rightarrow_{M_L} (i, 0, 0)\}.$$

Then $H_{M_L,1}$ is $N(FO)$ and $H_{M_L,2}$ is $F(FO)$.

Here halting of M_L means that the ID sequence becomes constant when reaching a stop state. This stop condition can be assumed without loss of generality [15].

4.1.3 Conservative reductions

Recall the following definitions from [1, 15].

Definition 4.2 [15]: Let X and Y be recursive classes of sentences. A *reduction* from X to Y is a recursive function $f : X \to Y$ such that for any $\psi \in X$, ψ is satisfiable iff $f(\psi)$ is satisfiable.

A *conservative reduction* from X to Y is a recursive function $f : X \to Y$ such that for any $\psi \in X$,

- ψ is satisfiable iff $f(\psi)$ is satisfiable; and
- ψ is finitely satisfiable iff $f(\psi)$ is finitely satisfiable.

A recursive class of sentences X is said to be a *conservative reduction class* if there exists a conservative reduction from FO to X. ∎

The *(finite) satisfiability problem* for a recursive class of sentences X is the problem of determining, given any $\psi \in X$, whether ψ has a (finite) model.

Obviously, if a recursive class of sentences X is a conservative reduction class, then

- the satisfiability problem for X is co-r.e. complete; and
- the finite satisfiability problem for X is r.e. complete.

To show that a recursive subset X of FO is a conservative reduction class, it suffices to reduce $N(FO)$ and $F(FO)$ to $N(X)$ and $F(X)$, respectively. More precisely, this is described by the notion of semi-conservative reductions as follows.

Definition 4.3 [15]: Let X and Y be recursive classes of sentences. A *semi-conservative reduction* from X to Y is a recursive function $f : X \to Y$ such that

- $f(N(X)) \subseteq N(Y)$; and
- $f(F(X)) \subseteq F(Y)$. ∎

Lemma 4.4 [15]: If there exists a semi-conservative reduction from FO to a recursive subset X of FO, then X is a conservative reduction class. ∎

4.2 The undecidability of the implication problems for P_+

Next, we establish the undecidability of the implication and finite implication problems for P_+ defined in Section 4.1.

We prove Theorem 4.2. It suffices to show that the set

$$S(P_+) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \varphi \in P_+,\ \Sigma \subset P_+,\ \Sigma \text{ is finite}\}$$

is a *conservative reduction class*. To establish the conservative reduction class property for $S(P_+)$, by Lemma 4.4, it suffices to show that there is a semi-conservative reduction from FO to $S(P_+)$.

We establish the existence of such a semi-conservative reduction by reduction from the halting problem for two-register machines. To do this, we first present an encoding of two-register machines in terms of sentences in P_+, and then prove a reduction property of the encoding. Using this reduction property, we define a semi-conservative reduction from FO to $S(P_+)$.

4.2.1 Encoding

Let M be a two-register machine. Assume that M is programmed by

$$I_0,\ I_1,\ \ldots,\ I_l.$$

We also assume that the set E of binary relation symbols in signature σ includes:

- the predicates encoding the states of M:
  - $K_0, K_1, ..., K_l$,
  - $K_0^-, K_1^-, ..., K_l^-$;
- the predicates encoding the contents of the registers (natural numbers):
  - R_1^+, R_1^-: to encode the successor and the predecessor of the contents of $register_1$;
  - R_2^+, R_2^-: to encode the successor and the predecessor of the contents of $register_2$;

- E_{01}, E_{01}^{-}: to indicate that the content of $register_1$ is 0;
- E_{02}, E_{02}^{-}: to indicate that the content of $register_2$ is 0;

- the predicates distinguishing $register_1$ from $register_2$ and identifying the root r:
  - L_1, L_1^{-}: to identify $register_1$;
  - L_2, L_2^{-}: to identify $register_2$; and
  - L_r: to identify the root r.

We now define the encoding as follows.

Registers

We encode the contents of the registers by Φ_N, which is the conjunction of the path constraints of P_+ given below.

- Successor, predecessor:
  - $\phi_1 = \forall\, x\,(L_1(r,x) \rightarrow \forall\, y\,(R_1^{+}(x,y) \rightarrow R_1^{-}(y,x)))$
  - $\phi_2 = \forall\, x\,(L_1(r,x) \rightarrow \forall\, y\,(R_1^{-}(x,y) \rightarrow R_1^{+}(y,x)))$
  - $\phi_3 = \forall\, x\,(L_2(r,x) \rightarrow \forall\, y\,(R_2^{+}(x,y) \rightarrow R_2^{-}(y,x)))$
  - $\phi_4 = \forall\, x\,(L_2(r,x) \rightarrow \forall\, y\,(R_2^{-}(x,y) \rightarrow R_2^{+}(y,x)))$

    (ϕ_1, ϕ_2, ϕ_3 and ϕ_4 are constraints of the backward form.)
  - $\phi_5 = \forall\, x\,(L_1(r,x) \rightarrow \exists\, y\,(R_1^{+}(x,y) \wedge L_1^{-}(y,r)))$
  - $\phi_6 = \forall\, x\,(L_2(r,x) \rightarrow \exists\, y\,(R_2^{+}(x,y) \wedge L_2^{-}(y,r)))$

    (ϕ_5 and ϕ_6 are simple constraints of the backward form.)

- Register identification:
  - $\phi_7 = \forall\, x\,(\exists\, y\,(L_1(r,y) \wedge R_1^{+}(y,x)) \rightarrow L_1(r,x))$
  - $\phi_8 = \forall\, x\,(\exists\, y\,(L_1(r,y) \wedge R_1^{-}(y,x)) \rightarrow L_1(r,x))$
  - $\phi_9 = \forall\, x\,(\exists\, y\,(L_2(r,y) \wedge R_2^{+}(y,x)) \rightarrow L_2(r,x))$

- $\phi_{10} = \forall\, x\, (\exists\, y\, (L_2(r,y) \wedge R_2^-(y,x)) \rightarrow L_2(r,x))$

  (ϕ_7, ϕ_8, ϕ_9 and ϕ_{10} are simple constraints of the forward form.)

- States: for $i \in [0,l]$,

  - $\phi_{11}^i = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (K_i(x,y) \rightarrow K_i^-(y,x)))$
  - $\phi_{12}^i = \forall\, x\, (L_2(r,x) \rightarrow \forall\, y\, (K_i^-(x,y) \rightarrow K_i(y,x)))$

    (ϕ_{11}^i and ϕ_{12}^i are constraints of the backward form.)

- Zeros:

  - $\phi_{13} = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (E_{01}^-(x,y) \rightarrow E_{01}(y,x)))$
  - $\phi_{14} = \forall\, x\, (\exists\, y\, (L_1(r,y) \wedge E_{01}^-(y,x)) \rightarrow L_r(r,x))$
  - $\phi_{15} = \forall\, x\, (\exists\, y\, (L_r(r,y) \wedge E_{01}(y,x)) \rightarrow E_{01}(r,x))$
  - $\phi_{16} = \forall\, x\, (\exists\, z\, (L_1(r,z) \wedge \exists\, y\, (E_{01}^-(z,y) \wedge E_{02}(y,x))) \rightarrow E_{02}(r,x))$

    (ϕ_{13} is a constraint of the backward form, ϕ_{14}, ϕ_{15} and ϕ_{16} are simple constraints of the forward form.)

  - $\phi_{17} = \forall\, x\, (E_{01}(r,x) \rightarrow L_1(r,x))$
  - $\phi_{18} = \forall\, x\, (E_{02}(r,x) \rightarrow L_2(r,x))$

    (ϕ_{17} and ϕ_{18} are simple constraints of the forward form.)

IDs

We encode each ID $C = (i,m,n)$ of M by

$$\varphi_C = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, ((R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(x,y) \rightarrow K_i(x,y))),$$

where $\alpha \cdot \beta$ (abbreviation for $\alpha(x,z) \cdot \beta(z,y)$) stands for the concatenation of paths $\alpha(x,z)$ and $\beta(z,y)$, and $(\alpha)^m$ stands for the m-time concatenations of α, as defined in Section 2.2.

The constraint φ_C is in P_+. It has the forward form. More specifically, $pf(\varphi_C) = L_1$, $lt(\varphi_C) = (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n$, and $rt(\varphi_C) = K_i$. The notations pf, lt and rt are described in Definition 2.2.

Instructions

For each $i \in [0, l]$, we encode the instruction I_i by ϕ_{I_i} given below.

- Addition:
  - For $(i, register_1, j)$, ϕ_{I_i} is

$$\phi^i_{a_1} = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (\exists\, x'\, (R_1^-(x,x') \wedge K_i(x',y)) \rightarrow K_j(x,y))).$$

    Here $\phi^i_{a_1}$ is a constraint of the forward form with $pf(\phi^i_{a_1}) = L_1$, $lt(\phi^i_{a_1}) = R_1^- \cdot K_i$ and $rt(\phi^i_{a_1}) = K_j$.
  - For $(i, register_2, j)$, ϕ_{I_i} is

$$\phi^i_{a_2} = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (\exists\, y'\, (K_i(x,y') \wedge R_2^+(y',y)) \rightarrow K_j(x,y))).$$

    Here $\phi^i_{a_2}$ is a constraint of the forward form with $pf(\phi^i_{a_2}) = L_1$, $lt(\phi^i_{a_2}) = K_i \cdot R_2^+$ and $rt(\phi^i_{a_2}) = K_j$.
- Subtraction:
  - For $(i, register_1, j, k)$, ϕ_{I_i} is $\phi^i_{s_1} = \phi^i_{s_{1,0}} \wedge \phi^i_{s_{1,n}}$, where

$$\begin{aligned}
\phi^i_{s_{1,0}} &= \forall\, x\, (E_{01}(r,x) \rightarrow \forall\, y\, (K_i(x,y) \rightarrow K_j(x,y))),\\
\phi^i_{s_{1,n}} &= \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (\exists\, x'\, (R_1^+(x,x') \wedge K_i(x',y)) \rightarrow K_k(x,y))).
\end{aligned}$$

    Here $\phi^i_{s_1}$ is a conjunction of forward two constraints. The first conjunct $\phi^i_{s_{1,0}}$ is a constraint with $pf(\phi^i_{s_{1,0}}) = E_{01}$, $lt(\phi^i_{s_{1,0}}) = K_i$ and $rt(\phi^i_{s_{1,0}}) = K_j$. In the second conjunct $\phi^i_{s_{1,n}}$, $pf(\phi^i_{s_{1,n}}) = L_1$, $lt(\phi^i_{s_{1,n}}) = R_1^+ \cdot K_i$ and $rt(\phi^i_{s_{1,n}}) = K_k$.
  - For $(i, register_2, j, k)$, ϕ_{I_i} is $\phi^i_{s_2} = \phi^i_{s_{2,0}} \wedge \phi^i_{s_{2,n}}$, where

$$\begin{aligned}
\phi^i_{s_{2,0}} &= \forall\, x\, (E_{02}(r,x) \rightarrow \forall\, y\, (K_i^-(x,y) \rightarrow K_j(y,x))),\\
\phi^i_{s_{2,n}} &= \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (\exists\, y'\, (K_i(x,y') \wedge R_2^-(y',y)) \rightarrow K_k(x,y))).
\end{aligned}$$

    Here $\phi^i_{s_2}$ is a conjunction of two constraints. The first conjunct is a constraint of the backward form with $pf(\phi^i_{s_{2,0}}) = L_1$, $lt(\phi^i_{s_{2,0}}) = K_i^-$ and $rt(\phi^i_{s_{2,0}}) = K_j$. The second conjunct is a constraint of the forward form with $pf(\phi^i_{s_{2,n}}) = L_1$, $lt(\phi^i_{s_{2,n}}) = K_i \cdot R_2^-$ and $rt(\phi^i_{s_{2,n}}) = K_k$.

The encoding of the program of M is $\Phi_M = \bigwedge_{i=0}^{l} \phi_{I_i}$. Clearly, Φ_M is a conjunction of path constraints in P_+.

Now we show that the encoding above has the following reduction property.

Proposition 4.5: Let M be a two-register machine. For all IDs C and D of M,

$$C \Rightarrow_M D \quad \text{iff} \quad \Phi_N \wedge \Phi_M \wedge \varphi_C \rightarrow \varphi_D \text{ is valid.}$$

■

Proof: The proof consists of two parts.

(1) Assume $C \Rightarrow_M D$. We show that for each model G of $\Phi_N \wedge \Phi_M \wedge \varphi_C$, $G \models \varphi_D$. To show this, it suffices to show that for each natural number t and each ID C' of M, if C' is reached by M in t steps starting from C (denoted by $C \Rightarrow_M^t C'$), then $G \models \varphi_{C'}$. We prove this claim by induction on t.

Base case: If $t = 0$, then the claim holds since $G \models \varphi_C$.

Inductive step: Assume the claim for t.

Suppose that $C \Rightarrow_M^t C_1 \rightarrow_M^{I_i} C'$, where $C_1 = (i, m, n)$, and $C_1 \rightarrow_M^{I_i} C'$ stands for that C' is reached by executing instruction I_i at C_1. Then by the induction hypothesis, we have $G \models \varphi_{C_1}$. That is

$$G \models \forall x \, (L_1(r, x) \rightarrow \forall y \, ((R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(x, y) \rightarrow K_i(x, y))).$$

We show by *reductio* that the claim holds for $t + 1$ for each case of I_i, which has six cases in total.

Case 1: $I_i = (i, register_1, j)$. In this case, C' must be $(j, m + 1, n)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^{m+1} \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, b) \wedge \neg K_j(a, b),$$

then there exists $c \in |G|$, such that

$$G \models R_1^-(a, c) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(c, b).$$

By ϕ_8 in Φ_N, we have $G \models L_1(r,c)$. Thus by $G \models \varphi_{C_1}$, $G \models K_i(c,b)$. Hence

$$G \models L_1(r,a) \wedge R_1^-(a,c) \wedge K_i(c,b).$$

Thus by $\phi_{a_1}^i$ in Φ_M, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 2: $I_i = (i, register_2, j)$. In this case, C' must be $(j, m, n+1)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^{n+1}(a,b) \wedge \neg K_j(a,b),$$

then there exists $c \in |G|$, such that

$$G \models (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a,c) \wedge R_2^+(c,b).$$

By $G \models \varphi_{C_1}$, we have $G \models K_i(a,c)$. Hence

$$G \models L_1(r,a) \wedge K_i(a,c) \wedge R_2^+(c,b).$$

Thus by $\phi_{a_2}^i$ in Φ_M, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 3: $I_i = (i, register_1, j, k)$ and $m = 0$. In this case, C' must be $(j, 0, n)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a,b) \wedge \neg K_j(a,b),$$

then by $G \models \varphi_{C_1}$, we have $G \models K_i(a,b)$. In addition, there exists $c \in |G|$, such that

$$G \models L_1(r,a) \wedge E_{01}^-(a,c).$$

By ϕ_{13}, ϕ_{14} and ϕ_{15} in Φ_N, we have $G \models E_{01}(r,a)$. Hence

$$G \models E_{01}(r,a) \wedge K_i(a,b).$$

Thus by $\phi_{s_{1,0}}^i$ in Φ_M, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 4: $I_i = (i, register_1, j, k)$ and $m = p + 1$. In this case, C' must be (k, p, n).

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^p \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a,b) \wedge \neg K_k(a,b),$$

then by ϕ_5 in Φ_N, there exists $c \in |G|$, such that

$$G \models L_1(r,a) \wedge R_1^+(a,c).$$

By ϕ_7, ϕ_1 in Φ_N, we have $G \models L_1(r,c) \wedge R_1^-(c,a)$. Hence

$$G \models L_1(r,c) \wedge (R_1^-)^{p+1} \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(c,b).$$

Thus by $G \models \varphi_{C_1}$, $G \models K_i(c,b)$. Hence

$$G \models L_1(r,a) \wedge R_1^+(a,c) \wedge K_i(c,b).$$

Thus by $\phi^i_{s_{1,n}}$ in Φ_M, we have $G \models K_k(a,b)$. This contradicts the assumption.

Case 5: $I_i = (i, register_2, j, k)$ and $n = 0$. In this case, C' must be $(j, m, 0)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02}(a,b) \wedge \neg K_j(a,b),$$

then by $G \models \varphi_{C_1}$, we have $G \models K_i(a,b)$. By ϕ^i_{11} in Φ_N, $G \models K_i^-(b,a)$. Moreover, there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a,d) \wedge E_{01}^-(d,c) \wedge E_{02}(c,b).$$

By $G \models L_1(r,a)$ and ϕ_8 in Φ_N, we have $G \models L_1(r,d)$. Thus by ϕ_{16} in Φ_N, $G \models E_{02}(r,b)$. Hence

$$G \models E_{02}(r,b) \wedge K_i^-(b,a).$$

Thus by $\phi^i_{s_{2,0}}$ in Φ_M, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 6: $I_i = (i, register_2, j, k)$ and $n = p + 1$. In this case, C' must be (k, m, p).

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^p(a,b) \wedge \neg K_k(a,b),$$

then there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a,c) \wedge E_{01}^- \cdot E_{02}(c,d) \wedge (R_2^+)^p(d,b).$$

By ϕ_8 in Φ_N, we have $G \models L_1(r,c)$. By ϕ_{16} in Φ_N, $G \models E_{02}(r,d)$. By ϕ_{18} in Φ_N, $G \models L_2(r,d)$. By ϕ_9 in Φ_N, $G \models L_2(r,b)$. Therefore, by ϕ_6 in Φ_N, there exists $e \in |G|$, such that $G \models R_2^+(b,e)$. Hence

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^{p+1}(a,e).$$

By $G \models \varphi_{C_1}$, we have $G \models K_i(a,e)$. By ϕ_3 in Φ_N and $G \models R_2^+(b,e)$, $G \models R_2^-(e,b)$. Hence

$$G \models L_1(r,a) \wedge K_i(a,e) \wedge R_2^-(e,b).$$

Thus by $\phi^i_{s_{2,n}}$ in Φ_M, we have $G \models K_k(a,b)$. This contradicts the assumption.

Hence the claim holds for $t+1$ for all the cases of I_i.

(2) Conversely, assume $C \not\Rightarrow_M D$. We show that $\Phi_N \wedge \Phi_M \wedge \varphi_C \rightarrow \varphi_D$ is not valid. That is, we show that $\Phi_N \wedge \Phi_M \wedge \varphi_C \wedge \neg\varphi_D$ is satisfiable. To show this, we construct a σ-structure G such that $G \models \Phi_N \wedge \Phi_M \wedge \varphi_C$ and $G \models \neg\varphi_D$.

The structure G is defined as follows. The universe of G consists of a distinguished node rt, which is the interpretation of the constant r in G, and two distinct infinite chains of natural numbers. More specifically, let $\mathbb{N}$ denote the set of all natural numbers, then

$$|G| = \{rt\} \cup \mathbb{N} \cup \{i' \mid i \in \mathbb{N}\}.$$

The relations in G are populated as follows (the superscript G is omitted in the relation names):

$$\begin{array}{lcllcl}
L_r & = & \{(rt, rt)\} & & & \\
E_{01} & = & \{(rt, 0)\} & & E_{01}^- & = & \{(0, rt)\} \\
E_{02} & = & \{(rt, 0')\} & & E_{02}^- & = & \{(0', rt)\} \\
L_1 & = & \{(rt, i) \mid i \in \mathbb{N}\} & & L_1^- & = & \{(i, rt) \mid i \in \mathbb{N}\} \\
L_2 & = & \{(rt, i') \mid i \in \mathbb{N}\} & & L_2^- & = & \{(i', rt) \mid i \in \mathbb{N}\}
\end{array}$$

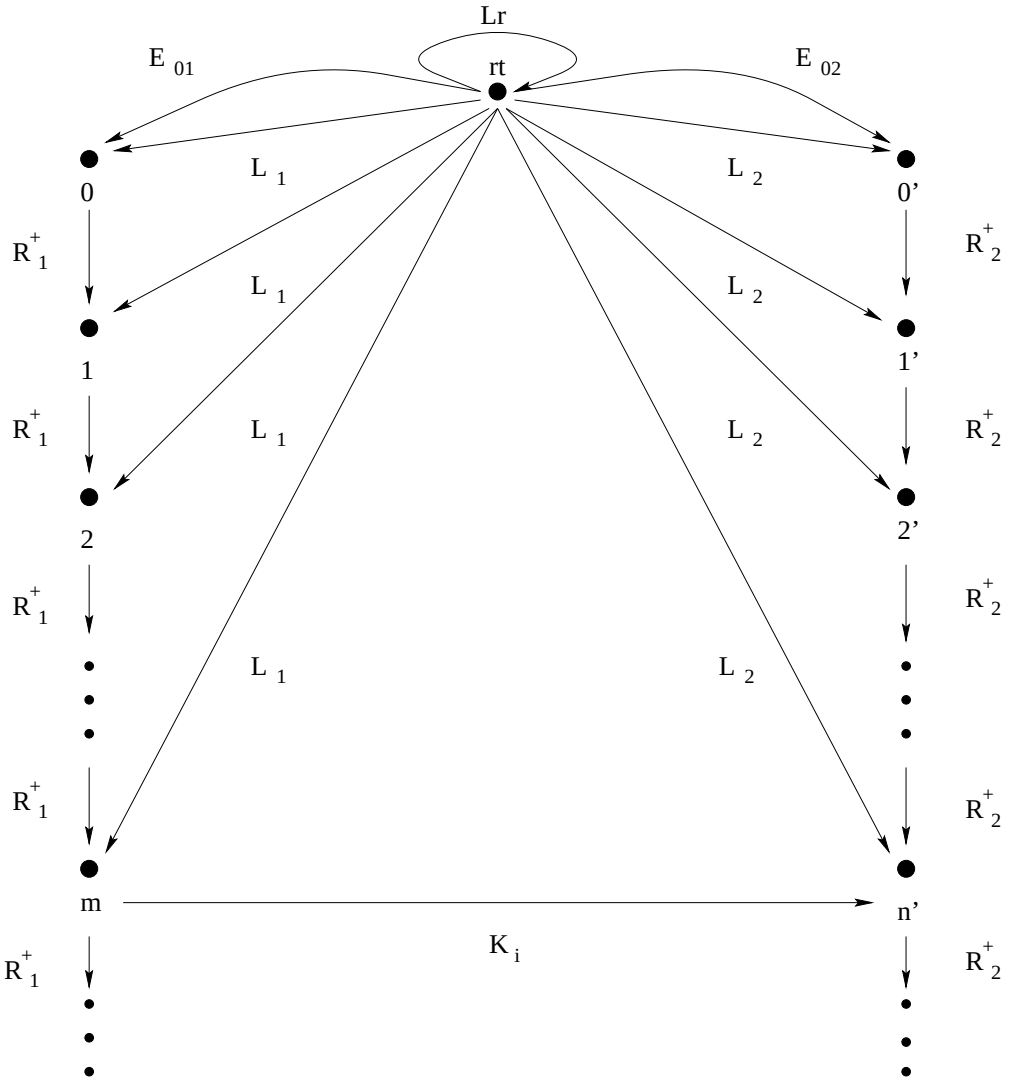

Figure 4.2: The structure G in Proposition 4.5

$$
\begin{array}{llll}
R_1^+ & = & \{(i, i+1) \mid i \in \mathbb{N}\} & R_1^- = \{(i+1, i) \mid i \in \mathbb{N}\} \\
R_2^+ & = & \{(i', (i+1)') \mid i \in \mathbb{N}\} & R_2^- = \{((i+1)', i') \mid i \in \mathbb{N}\} \\
K_i & = & \{(m, n') \mid C \Rightarrow_M (i, m, n)\} & K_i^- = \{(n', m) \mid (m, n') \in K_i\}
\end{array}
$$

See Figure 4.2 for the structure G ($E_{01}^-, E_{02}^-, L_1^-, L_2^-, R_1^-, R_2^-, K_i^-$ edges are omitted in the graph).

It is easy to verify the following claims.

Claim 1: $G \models \varphi_C \wedge \neg\varphi_D$.

This is because $C \Rightarrow_M C$ and $C \not\Rightarrow_M D$.

Claim 2: $G \models \Phi_N$.

This is immediate from the construction of G.

Claim 3: $G \models \Phi_M$.

Claim 3 follows from the simple facts given below.

- *Fact 1*: $G \models K_i(m, n')$ iff $C \Rightarrow_M (i, m, n)$.
- *Fact 2*: If $C \Rightarrow_M (i, m, n) \rightarrow_M^{I_i} C'$, then $C \Rightarrow_M C'$. Moreover, C' must satisfy the following conditions.
  - If $I_i = (i, register_1, j)$, then $C' = (j, m + 1, n)$.
  - If $I_i = (i, register_2, j)$, then $C' = (j, m, n + 1)$.
  - If $I_i = (i, register_1, j, k)$ and $m = 0$, then $C' = (j, 0, n)$.
  - If $I_i = (i, register_1, j, k)$ and $m = p + 1$, then $C' = (j, p, n)$.
  - If $I_i = (i, register_2, j, k)$ and $n = 0$, then $C' = (j, m, 0)$.
  - If $I_i = (i, register_2, j, k)$ and $n = p + 1$, then $C' = (j, m, p)$.
- *Fact 3*: If $G \not\models \Phi_M$, i.e., there exists I_i for some $i \in [0, l]$ such that $G \not\models \phi_{I_i}$, then there exist $m, n' \in |G|$, such that
  - if $I_i = (i, register_1, j)$, then $G \models K_i(m, n') \wedge \neg K_j(m + 1, n')$,
  - if $I_i = (i, register_2, j)$, then $G \models K_i(m, n') \wedge \neg K_j(m, (n + 1)')$,
  - if $I_i = (i, register_1, j, k)$, then either
    - $G \models K_i(0, n') \wedge \neg K_j(0, n')$, where $m = 0$, or
    - $G \models K_i(p + 1, n') \wedge \neg K_k(p, n')$, where $m = p + 1$,
  - if $I_i = (i, register_2, j, k)$, then either
    - $G \models K_i(m, 0') \wedge \neg K_j(m, 0')$, where $n = 0$, or
    - $G \models K_i(m, (p + 1)') \wedge \neg K_k(m, p')$, where $n = p + 1$.

Using these facts, Claim 3 can be easily verified by *reductio*. More specifically, suppose $G \not\models \Phi_M$. Then there is $i \in [0, l]$ such that $G \not\models \phi_{I_i}$. Here I_i has six cases. For each of these cases, the assumption contradicts the facts above. As an example, consider the case where I_i is $(i, register_1, j)$. By Fact 3, there must be nodes $m, n' \in |G|$, such that $G \models K_i(m, n') \wedge \neg K_j(m+1, n')$. By Fact 1, $C \Rightarrow_M (i, m, n')$. In addition, by Fact 2, we have $C \Rightarrow_M (j, m+1, n')$. Thus again by Fact 1, $G \models K_j(m+1, n')$. This contradicts the assumption. The proofs for the other cases are similar.

Therefore, if $C \not\Rightarrow_M D$, then $\Phi_N \wedge \Phi_M \wedge \varphi_C \wedge \neg\varphi_D$ is satisfiable.

This completes the proof of Proposition 4.5. ∎

4.2.2 Semi-conservative reduction

Taking advantage of the reduction property established above, we now define a recursive function $f : FO \to S(P_+)$ by:

$$f(\psi) \quad \mapsto \quad \Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$$

where $C(\psi)$ is the ID $(0, m(\psi), 0)$ of M_L with an appropriate encoding $m(\psi)$ of ψ, as described in Section 4.1.2.

The proposition below shows that f is indeed a semi-conservative reduction from FO to $S(P_+)$.

Proposition 4.6: Let M_L be the two-register machine described in Section 4.1.2. For each first-order sentence ψ,

1. $\psi \in H_{M_L,1}$ iff $f(\psi)$ is not satisfiable; and
2. if $\psi \in H_{M_L,2}$, then $f(\psi)$ has a finite model.

∎

Proof: Recall $H_{M_L,1} = N(FO)$ and $H_{M_L,2} = F(FO)$ from Section 4.1.2.

(1) By Proposition 4.5, we have that

$$C(\psi) \Rightarrow_{M_L} (1, 0, 0) \quad \text{iff} \quad \Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \to \varphi_{(1,0,0)} \text{ is valid.}$$

That is, $C(\psi) \Rightarrow_{M_L} (1,0,0)$ iff $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$ is not satisfiable. Notice that $\psi \in H_{M_L,1}$ iff $C(\psi) \Rightarrow_{M_L} (1,0,0)$. Therefore, $\psi \in H_{M_L,1}$ iff $f(\psi)$ is not satisfiable. This completes the proof of claim 1.

(2) We show that if $\psi \in H_{M_L,2}$, then $f(\psi)$ has a finite model.

First note that if $\psi \in H_{M_L,2}$, then the computation of M_L with initial ID $C(\psi)$ is finite. Therefore, the set

$$SID_{C(\psi)} \;=\; \{(i,m,n) \mid C(\psi) \Rightarrow_{M_L} (i,m,n)\}$$

is finite. Hence there is a natural number p, such that for each $(i,m,n) \in SID_{C(\psi)}$, $m+2 \leq p$ and $n+2 \leq p$.

Now we construct a finite model H for $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$. The universe of H has $2p+1$ nodes. More specifically,

$$|H| \;=\; \{rt, 1, 2, ..., p\} \;\cup\; \{1', 2', ..., p'\},$$

where rt is the interpretation of the constant r in H.

The binary relations $L_r, E_{01}, E_{02}, E_{01}^-, E_{02}^-, K_i$ and K_i^- in H are exactly the same as those in the σ-structure G given in the proof of Proposition 4.5. The binary relations L_1, L_1^-, L_2, L_2^-, R_1^+, R_1^-, R_2^+ and R_2^- are populated in H as follows (the superscript H is omitted in the relation names):

$$\begin{aligned}
R_1^+ &= \{(i,i+1) \mid 0 \leq i < p\} \cup \{(p,p)\} \\
R_1^- &= \{(i+1,i) \mid 0 \leq i < p\} \cup \{(p,p)\} \\
R_2^+ &= \{(i',(i+1)') \mid 0 \leq i < p\} \cup \{(p',p')\} \\
R_2^- &= \{((i+1)',i') \mid 0 \leq i < p\} \cup \{(p',p')\} \\
L_1 &= \{(rt,i) \mid 0 \leq i \leq p\} \\
L_1^- &= \{(i,rt) \mid 0 \leq i \leq p\} \\
L_2 &= \{(rt,i') \mid 0 \leq i \leq p\} \\
L_2^- &= \{(i',rt) \mid 0 \leq i \leq p\}
\end{aligned}$$

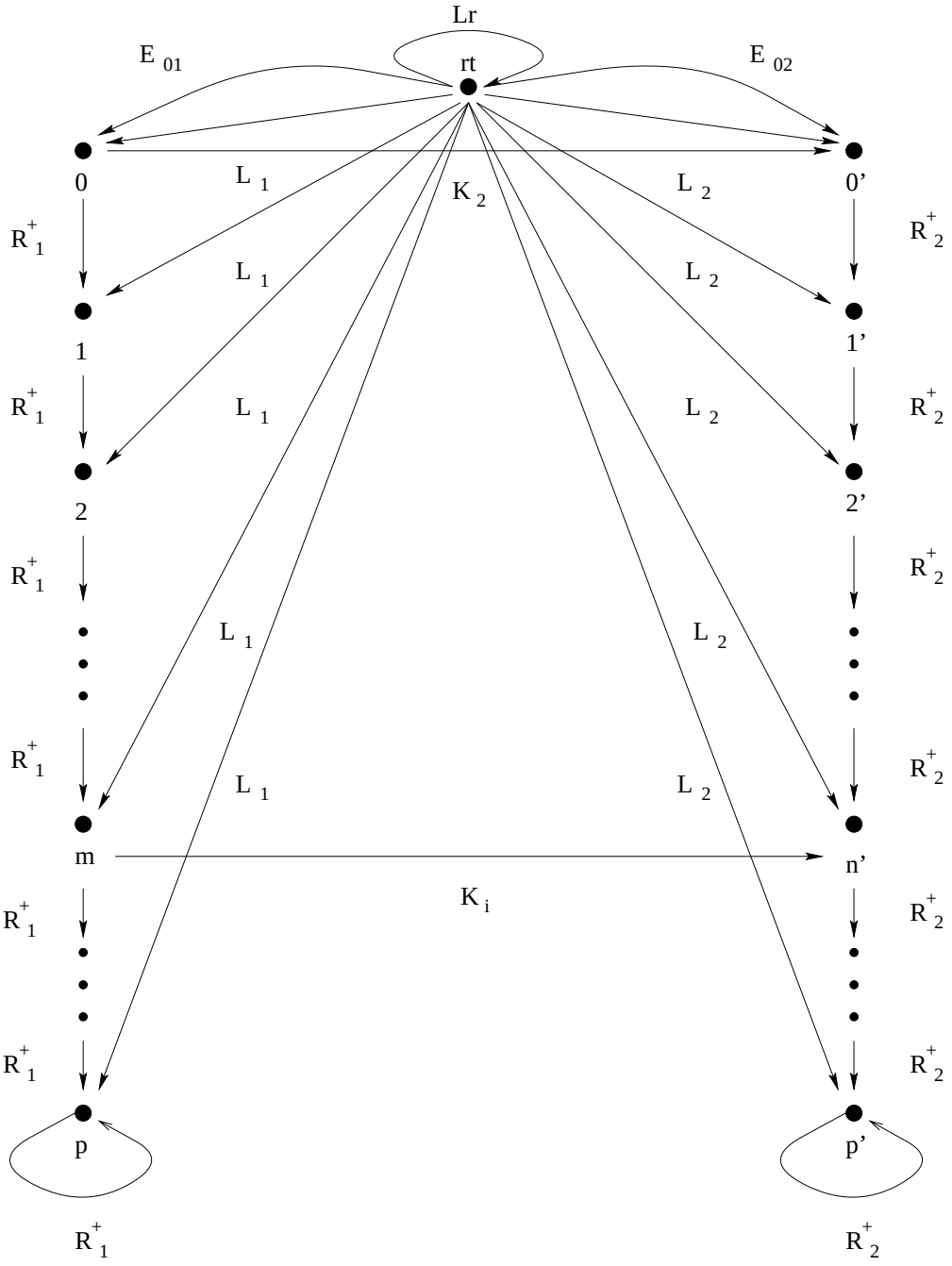

Figure 4.3: The structure H in Proposition 4.6

See Figure 4.3 for the structure H ($E_{01}^-, E_{02}^-, L_1^-, L_2^-, R_1^-, R_2^-, K_i^-$ edges are omitted in the graph).

Note that the relations K_i and K_i^- in H are well-defined, since if $C(\psi) \Rightarrow_{M_L} (i, m, n)$, then $m < p - 1$ and $n < p - 1$. In addition, it is easy to verify that H is well-defined.

We now show that H is indeed a model of $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

First, by $C(\psi) \Rightarrow_{M_L} C(\psi)$ and $C(\psi) \not\Rightarrow_{M_L} (1, 0, 0)$, we have that $H \models \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

Second, it is easy to verify that $H \models \Phi_N$. Note here it is to ensure $H \models \phi_5 \wedge \phi_6$ that we require $H \models R_1^+(p, p) \wedge R_2^+(p', p')$.

Finally, we show that $H \models \Phi_M$. It is straightforward to verify the following simple facts.

- *Fact 4:* If $C(\psi) \Rightarrow_{M_L} (i, m, n)$, then $m < p - 1$ and $n < p - 1$.
- *Fact 5:* If $(i, m, n) \rightarrow_{M_L}^{I_i} (j, m_1, n_1)$, then $m_1 \leq m + 1$ and $n_1 \leq n + 1$. As a result of Fact 4, $m_1 < p$ and $n_1 < p$. Here $(i, m, n) \rightarrow_{M_L}^{I_i} (j, m_1, n_1)$ stands for that M_L reaches (j, m_1, n_1) by executing instruction I_i at (i, m, n).

In addition, Facts 1, 2 and 3 for showing $G \models \Phi_M$ in the proof of Proposition 4.5 are also true here. Therefore, the argument for showing $G \models \Phi_M$ in the proof of Proposition 4.5, together with Facts 4 and 5 above, proves $H \models \Phi_M$.

Hence H is indeed a finite model of $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

This completes the proof of Proposition 4.6. ∎

Corollary 4.7: The function f defined above is a reduction from FO to $S(P_+)$. ∎

Proof: By the definition of M_L, for each $\psi \in FO$, ψ is satisfiable iff $\psi \notin H_{M_L,1}$. As shown in the proof of Proposition 4.6, $\psi \notin H_{M_L,1}$ iff $f(\psi)$ is satisfiable. Therefore, f is a reduction from FO to $S(P_+)$. ∎

As an immediate result of Proposition 4.6 and Lemma 4.4, we have the following corollary.

Corollary 4.8: The set $S(P_+)$ is a conservative reduction class. ∎

From Corollary 4.8, Theorem 4.2 follows immediately.

4.3 The undecidability of the implication problems for P_f

Finally, we establish Theorem 4.3. As in the proof of Theorem 4.2, we show that the set

$$S(P_f) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \varphi \in P_f,\ \Sigma \subset P_f,\ \Sigma \text{ is finite}\}$$

is a *conservative reduction class*. To do this, we first present an encoding of two-register machines with sentences in P_f, and then prove a reduction property of the encoding. Using this reduction property, we define a semi-conservative reduction from FO to $S(P_f)$.

4.3.1 Encoding

Let M be a two-register machine as described in Section 4.2.1. Assume that the set E of binary relation symbols in signature σ is the same as the one described in Section 4.2.1, except that the predicates L_r and K_i^- for $i \in [0, l]$ are no longer required here.

We define the encoding as follows.

Registers

We encode the contents of the registers by Φ_N^f, which is the conjunction of the path constraints of P_f given below.

- Successor, predecessor:
  - $\phi_1 = \forall\, x\,(L_1(r,x) \rightarrow \forall\, y\,(\exists\, z\,(R_1^+(x,z) \wedge R_1^-(z,y)) \rightarrow \epsilon(x,y)))$
    $(pf(\phi_1) = L_1,\ lt(\phi_1) = R_1^+ \cdot R_1^-$ and $rt(\phi_1) = \epsilon.)$
  - $\phi_2 = \forall\, x\,(L_1(r,x) \rightarrow \forall\, y\,(\exists\, z\,(R_1^-(x,z) \wedge R_1^+(z,y)) \rightarrow \epsilon(x,y)))$
  - $\phi_3 = \forall\, x\,(L_2(r,x) \rightarrow \forall\, y\,(\exists\, z\,(R_2^+(x,z) \wedge R_2^-(z,y)) \rightarrow \epsilon(x,y)))$
  - $\phi_4 = \forall\, x\,(L_2(r,x) \rightarrow \forall\, y\,(\exists\, z\,(R_2^-(x,z) \wedge R_2^+(z,y)) \rightarrow \epsilon(x,y)))$
  - $\phi_5 = \forall\, x\,(L_1(r,x) \rightarrow \forall\, y\,(\epsilon(x,y) \rightarrow \exists\, z\,(R_1^+(x,z) \wedge R_1^-(z,y))))$
    $(pf(\phi_5) = L_1,\ lt(\phi_5) = \epsilon$ and $rt(\phi_5) = R_1^+ \cdot R_1^-.)$
  - $\phi_6 = \forall\, x\,(L_2(r,x) \rightarrow \forall\, y\,(\epsilon(x,y) \rightarrow \exists\, z\,(R_2^+(x,z) \wedge R_2^-(z,y))))$

- Register identification: ϕ_7, ϕ_8, ϕ_9 and ϕ_{10} are the same as given in Section 4.2.1.
- Zeros:
  - $\phi_{11} = \forall\, x\, (\exists\, y\, (L_1(r,y) \wedge E_{01}^-(y,x)) \rightarrow \epsilon(r,x))$

    (ϕ_{11} is a simple path constraint with $lt(\phi_{11}) = L_1 \cdot E_{01}^-$ and $rt(\phi_{11}) = \epsilon$.)
  - $\phi_{12} = \forall\, x\, (L_1(r,x) \rightarrow$
    $\forall\, y\, (E_{01}^-(x,y) \rightarrow \exists\, z\, (E_{01}^-(x,z) \wedge \exists\, z'\, (E_{01}(z,z') \wedge E_{01}^-(z',y)))))$

    ($pf(\phi_{12}) = L_1$, $lt(\phi_{12}) = E_{01}^-$ and $rt(\phi_{12}) = E_{01}^- \cdot E_{01} \cdot E_{01}^-$.)
  - $\phi_{13} = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (\exists\, z\, (E_{01}^-(x,z) \wedge E_{01}(z,y)) \rightarrow \epsilon(x,y)))$

    ($pf(\phi_{13}) = L_1$, $lt(\phi_{13}) = E_{01}^- \cdot E_{01}$ and $rt(\phi_{13}) = \epsilon$.)
  - $\phi_{14} = \forall\, x\, (\exists\, y\, (L_1(r,y) \wedge \exists\, z\, (E_{01}^-(y,z) \wedge E_{02}(z,x))) \rightarrow E_{02}(r,x))$

    (ϕ_{14} is a simple constraint with $lt(\phi_{14}) = L_1 \cdot E_{01}^- \cdot E_{02}$ and $rt(\phi_{14}) = E_{02}$.)
  - $\phi_{15} = \forall\, x\, (E_{02}(r,x) \rightarrow \forall\, y\, (\epsilon(x,y) \rightarrow \exists\, z\, (E_{02}^-(x,z) \wedge E_{02}(z,y))))$

    ($pf(\phi_{15}) = E_{02}$, $lt(\phi_{15}) = \epsilon$ and $rt(\phi_{15}) = E_{02}^- \cdot E_{02}$.)
  - $\phi_{16} = \forall\, x\, (E_{02}(r,x) \rightarrow L_2(r,x))$

IDs

The encoding of each ID C of M, φ_C, is the same as the one given in Section 4.2.1.

Note that φ_C is in P_f.

Instructions

The encoding of each instruction I_i, ϕ_{I_i}, is the same as the one given in Section 4.2.1, except

$$\phi_{s_{2,0}}^i = \forall\, x\, (L_1(r,x) \rightarrow \forall\, y\, (K_i \cdot E_{02}^- \cdot E_{02}(x,y) \rightarrow K_j(x,y))).$$

Here $pf(\phi_{s_{2,0}}^i) = L_1$, $lt(\phi_{s_{2,0}}^i) = K_i \cdot E_{02}^- \cdot E_{02}$, and $rt(\phi_{s_{2,0}}^i) = K_j$.

The encoding of the program of M is $\Phi_M^f = \bigwedge_{i=0}^{l} \phi_{I_i}$.

It is clear that Φ_M^f is a conjunction of path constraints in P_f.

Analogous to Proposition 4.5, we establish the reduction property of the encoding above as follows.

Proposition 4.9: Let M be a two-register machine. For all IDs C and D of M, we have that

$$C \Rightarrow_M D \quad \text{iff} \quad \Phi_N^f \wedge \Phi_M^f \wedge \varphi_C \rightarrow \varphi_D \text{ is valid.}$$

■

Proof:

(1) Assume that $C \Rightarrow_M D$. As in the proof of Proposition 4.5, we prove by induction on step t that for each ID C' of M and each model G of $\Phi_N^f \wedge \Phi_M^f \wedge \varphi_C$, if $C \Rightarrow_M^t C'$ then $G \models \varphi_{C'}$. This can be shown in basically the same way as for Proposition 4.5, except for the following cases in the inductive step.

Case 3: $I_i = (i, register_1, j, k)$ and $m = 0$. In this case, C' must be $(j, 0, n)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, b) \wedge \neg K_j(a, b).$$

Then by $G \models \varphi_{C_1}$, we have $G \models K_i(a, b)$. In addition, there exists $e \in |G|$, such that

$$G \models L_1(r, a) \wedge E_{01}^-(a, e).$$

By ϕ_{12} in Φ_N^f, there exist $c, d \in |G|$, such that

$$G \models L_1(r, a) \wedge E_{01}^-(a, c) \wedge E_{01}(c, d).$$

Thus by ϕ_{13} in Φ_N^f, we have $G \models \epsilon(a, d)$. Hence

$$G \models L_1(r, a) \wedge E_{01}^-(a, c) \wedge E_{01}(c, a).$$

By $G \models L_1(r, a) \wedge E_{01}^-(a, c)$ and ϕ_{11} in Φ_N^f, we have $G \models \epsilon(r, c)$. Thus $G \models E_{01}(r, a)$. Hence

$$G \models E_{01}(r, a) \wedge K_i(a, b).$$

Thus by $\phi_{s_{1,0}}^i$ in Φ_M^f, we have $G \models K_j(a, b)$. This contradicts the assumption.

Case 4: $I_i = (i, register_1, j, k)$ and $m = p + 1$. In this case, C' must be (k, p, n).

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^p \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, b) \wedge \neg K_k(a, b).$$

Then by ϕ_5 in Φ_N^f, there exists $c \in |G|$, such that

$$G \models L_1(r, a) \wedge R_1^+(a, c) \wedge R_1^-(c, a).$$

By ϕ_7 in Φ_N^f, we have $G \models L_1(r, c) \wedge R_1^-(c, a)$. Hence

$$G \models L_1(r, c) \wedge (R_1^-)^{p+1} \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(c, b).$$

Thus by $G \models \varphi_{C_1}$, we have $G \models K_i(c, b)$. Hence

$$G \models L_1(r, a) \wedge R_1^+(a, c) \wedge K_i(c, b).$$

Thus by $\phi^i_{s_{1,n}}$ in Φ_M^f, we have $G \models K_k(a, b)$. This contradicts the assumption.

Case 5: $I_i = (i, register_2, j, k)$ and $n = 0$. In this case, C' must be $(j, m, 0)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02}(a, b) \wedge \neg K_j(a, b).$$

Then by $G \models \varphi_{C_1}$, we have $G \models K_i(a, b)$. Moreover, there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a, d) \wedge E_{01}^-(d, c) \wedge E_{02}(c, b).$$

By $G \models L_1(r, a)$ and ϕ_8 in Φ_N^f, we have $G \models L_1(r, d)$. Thus by ϕ_{14} in Φ_N^f, we have

$$G \models E_{02}(r, b).$$

By ϕ_{15} in Φ_N^f, there exists $e \in |G|$, such that

$$G \models E_{02}^-(b, e) \wedge E_{02}(e, b).$$

Hence

$$G \models L_1(r, a) \wedge K_i(a, b) \wedge E_{02}^-(b, e) \wedge E_{02}(e, b).$$

Thus by $\phi^i_{s_{2,0}}$ in Φ^f_M, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 6: $I_i = (i, register_2, j, k)$ and $n = p+1$. In this case, C' must be (k, m, p).

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^p(a,b) \wedge \neg K_k(a,b).$$

Then there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a,c) \wedge E_{01}^- \cdot E_{02}(c,d) \wedge (R_2^+)^p(d,b).$$

By ϕ_8 in Φ^f_N, we have $G \models L_1(r,c)$. By ϕ_{14} in Φ^f_N, $G \models E_{02}(r,d)$. By ϕ_{16} in Φ^f_N, $G \models L_2(r,d)$. By ϕ_9 in Φ^f_N, $G \models L_2(r,b)$. Therefore, by ϕ_6 in Φ^f_N, there exists $e \in |G|$, such that

$$G \models R_2^+(b,e) \wedge R_2^-(e,b).$$

Hence

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^{p+1}(a,e).$$

By $G \models \varphi_{C_1}$, we have $G \models K_i(a,e)$. Hence

$$G \models L_1(r,a) \wedge K_i(a,e) \wedge R_2^-(e,b).$$

Thus by $\phi^i_{s_{2,n}}$ in Φ^f_M, we have $G \models K_k(a,b)$. This contradicts the assumption.

(2) Conversely, assume that $C \not\Rightarrow_M D$. It is easy to verify that the structure G (without L_r and K_i^- edges) given in the proof of Proposition 4.5 is a model of $\Phi^f_N \wedge \Phi^f_M \wedge \varphi_C \wedge \neg\varphi_D$.

■

4.3.2 Semi-conservative reduction

We define a recursive function $g: \ FO \to S(P_f)$ by:

$$g(\psi) \quad \mapsto \quad \Phi^f_N \wedge \Phi^f_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$$

where $C(\psi)$ is the ID $(0, m(\psi), 0)$ of M_L with an appropriate encoding $m(\psi)$ of ψ, as described in Section 4.1.2.

Proposition 4.10 below shows that the function g is indeed a semi-conservative reduction from FO to $S(P_f)$.

Proposition 4.10: Let M_L be the two-register machine described in Section 4.1.2. For each first-order sentence ψ,

1. $\psi \in H_{M_L,1}$ iff $g(\psi)$ is not satisfiable; and
2. if $\psi \in H_{M_L,2}$, then $g(\psi)$ has a finite model.

■

Proof: The proof is the same as the proof of Proposition 4.6, except that here in the structure H shown in Figure 4.3, there are no L_r and K_i^- edges. ■

Corollary 4.11: The function g defined above is a reduction from FO to $S(P_f)$. ■

From Proposition 4.10 and Lemma 4.4 follows the corollary below. As a result, Theorem 4.3 follows.

Corollary 4.12: The set $S(P_f)$ is a conservative reduction class. ■

Chapter 5

Decidable Restricted Implication Problems in $\mathcal{SM}$

The undecidability results of Chapter 4 suggest that we search for fragments of P_c which possess decidable implication problems, and yet retain sufficient expressive power of the full language. This chapter identifies several fragments of P_c which share the following properties. First, they each properly contain the set of word constraints. Second, each of them fails to be included in two-variable first-order logic. Third, they allow the formulation of many semantic relations which are of interest from the point of view of database theory. And finally, the implication and finite implication problems for each of them are decidable in the context of $\mathcal{SM}$.

This chapter begins by introducing three fragments of P_c in Section 5.1. The decidability of the implication and finite implication problems associated with these fragments is established in Sections 5.2, 5.3 and 5.4, respectively. Finally, a mild generalization of P_c, $P_c^{\wedge}$, is investigated in Section 5.5.

5.1 Decidable fragments of P_c

In this section, we describe three fragments of P_c and their associated implication and finite implication problems.

5.1.1 The prefix restricted implication for P_c

The implication problems for simple path constraints, which are known to be decidable, can be viewed as a restricted form of the implication problems for P_c. More specifically, the implication problems for P_s are the implication problems for P_c under the following

restriction: for any finite subset of P_c in the implication problems, the prefix of each constraint in the subset is the empty path.

By replacing this prefix restriction with a weaker one, we define the prefix restricted implication problems for P_c as follows.

Definition 5.1: A *prefix restricted subset* of P_c is a finite subset of P_c in which the prefixes of all the constraints have the same length.

The *prefix restricted (finite) implication problem for* P_c is the problem of determining, given any prefix restricted subset $\Sigma \cup \{\varphi\}$ of P_c, whether all the (finite) models of Σ are also models of φ. In particular, in the context of $\mathcal{SM}$, that is the problem to determine whether for all (finite) σ-structure G, if $G \models \Sigma$ then $G \models \varphi$. ∎

Obviously, the implication problems for word constraints are special cases of the prefix restricted implication problems for P_c. Moreover, in contrast to word constraint implication, prefix restricted implication cannot be stated in two-variable first-order logic. A convenient argument for this is that $\{\varphi\}$, where φ is the constraint given in Example 4.1, is a prefix restricted subset of P_c. However, φ is not expressible in FO^2.

Many cases of integrity constraint implication commonly found in databases are instances of the prefix restricted implication problem for P_c. Among these are implications for inverse constraints and local database constraints. As an example, consider the set consisting of the following local inverse constraints in the school databases described in Chapter 1:

$$\forall s\,(\exists d\,(Schools(r,d) \wedge Students(d,s)) \rightarrow \forall c\,(Taking(s,c) \rightarrow Enrolled(c,s)))$$
$$\forall c\,(\exists d\,(Schools(r,d) \wedge Courses(d,c)) \rightarrow \forall s\,(Enrolled(c,s) \rightarrow Taking(s,c)))$$

and the constraint

$$\forall s_1\,(\exists d\,(Schools(r,d) \wedge Students(d,s_1)) \rightarrow \forall s_2\,(\epsilon(s_1,s_2) \rightarrow \exists c\,(Taking(s_1,c) \wedge Enrolled(c,s_2)))).$$

This set is a prefix restricted subset of P_c.

Another instance of prefix restricted implication is the implication of the constraint

$$\forall x\,(cities(r,x) \rightarrow \forall y\,(\exists z\,(connect(x,z) \wedge connect(z,y)) \rightarrow connect(y,x)))$$

from:

$$\forall x\,(cities(r,x) \rightarrow \forall y\,(\exists z\,(connect(x,z) \wedge connect(z,y)) \rightarrow connect(x,y)))$$
$$\forall x\,(cities(r,x) \rightarrow \forall y\,(connect(x,y) \rightarrow connect(y,x)))$$

5.1.2 Sublanguage P_β

Some cases of path constraint implication canvassed earlier are not instances of the prefix restricted implication. For instance, recall the two extent constraints and the two inverse constraints for student/course databases given in Chapter 1:

$$\forall c\,(\exists s\,(Students(r,s) \wedge Taking(s,c)) \rightarrow Courses(r,c))$$
$$\forall s\,(\exists c\,(Courses(r,c) \wedge Enrolled(c,s)) \rightarrow Students(r,s))$$

$$\forall s\,(Students(r,s) \rightarrow \forall c\,(Taking(s,c) \rightarrow Enrolled(c,s)))$$
$$\forall c\,(Courses(r,c) \rightarrow \forall s\,(Enrolled(c,s) \rightarrow Taking(s,c)))$$

The set consisting of these constraints is not a prefix restricted subset of P_c.

The constraints in the last example, however, are in the sublanguage P_β of P_c defined below.

Definition 5.2: A *β-restricted path constraint* φ is a constraint in P_c with $|lt(\varphi)| \leq 1$. That is, either $lt(\varphi) = \epsilon$, or $lt(\varphi) = K$ for some $K \in E$. Here the notation $lt(\varphi)$ is described in Definition 2.2.

The set of all simple path constraints and all β-restricted path constraints is denoted by P_β.

The *(finite) implication problem for P_β* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of P_β, whether all the (finite) models of Σ are also models of φ. In the

context of $\mathcal{SM}$, that is the problem to determine whether for all (finite) σ-structure G, if $G \models \Sigma$ then $G \models \varphi$. ∎

Note that the class of word constraints is a proper subset of P_β. In addition, not all constraints in P_β are expressible in two-variable first-order logic. Indeed, the constraint φ given in Example 4.1 is in P_β, but is not in FO^2.

5.1.3 The extended implication for P_β

Recall the local extent constraints given in Chapter 1:

$$\forall d \ \ (Schools(r,d) \rightarrow \forall c \ (\exists s \ (Students(d,s) \wedge Taking(s,c)) \rightarrow Courses(d,c)))$$
$$\forall d \ \ (Schools(r,d) \rightarrow \forall s \ (\exists c \ (Courses(d,c) \wedge Enrolled(c,s)) \rightarrow Students(d,s)))$$

Consider the set consisting of these local extent constraints and the local inverse constraints given in Section 5.1.1. This set is neither a prefix restricted subset of P_c nor a subset of P_β. However, the constraints in this set share the following property: all of them are constraints in student/course databases as shown in Figure 1.1 augmented with a common prefix `Schools`. In general, when represented in a global environment, path constraints in a local database are augmented with a common prefix.

This example motivates the following extension of P_β.

Definition 5.3: Let α be a path and φ be a constraint in P_β. The *extension of φ with prefix α*, denoted by $\delta(\varphi, \alpha)$, is the constraint defined either by

$$\forall x \ (\alpha \cdot pf(\varphi)(r,x) \rightarrow \forall y \ (lt(\varphi)(x,y) \rightarrow rt(\varphi)(x,y)))$$

when φ is of the forward form, or by

$$\forall x \ (\alpha \cdot pf(\varphi)(r,x) \rightarrow \forall y \ (lt(\varphi)(x,y) \rightarrow rt(\varphi)(y,x)))$$

when φ is of the backward form. Here $\cdot$ is the path concatenation operator, and pf, lt and rt are defined in Definition 2.2.

Let α be a path and Σ be a finite subset of P_β. The *extension of* Σ *with prefix* α is the subset of P_c defined by

$$\{\delta(\varphi, \alpha) \mid \varphi \in \Sigma\}.$$

Such a set is called a *prefix extended subset of* P_β.

The *extended (finite) implication problem for* P_β is the problem of determining, given any prefix extended subset $\Sigma \cup \{\varphi\}$ of P_β, whether all the (finite) models of Σ are also models of φ. In the context of $\mathcal{SM}$, that is the problem to determine whether for all (finite) σ-structure G, if $G \models \Sigma$ then $G \models \varphi$. ∎

For instance, the set described in the last example is a prefix extended subset of P_β.

Note that the (finite) implication problem for P_β is a special case of the extended (finite) implication problem for P_β. As an immediate result, implications of word constraints are special cases of extended implications of constraints of P_β. Moreover, extended implications of constraints of P_β cannot be stated in two-variable first-order logic.

5.2 Decidability of the prefix restricted implication for P_c

In this section, we establish the decidability of the prefix restricted implication problems for P_c.

Theorem 5.1: In the context of $\mathcal{SM}$, the prefix restricted implication and finite implication problems for P_c are decidable. ∎

The idea of the proof is to show that the satisfiability and finite satisfiability problems for the set

$$S_p = \{\bigwedge \Sigma \wedge \neg\varphi \mid \Sigma \cup \{\varphi\} \text{ is a prefix restricted subset of } P_c\}$$

are decidable. That is, we show that it is decidable to determine, given any $\psi \in S_p$, whether there is a (finite) σ-structure such that $G \models \psi$.

Recall the following notion from [15].

Definition 5.4 [15]: A recursive class X of first-order logic sentences has the *small model*

property for satisfiability iff there exists a recursive function s such that for each $\psi \in X$, if ψ is satisfiable, then ψ has a finite model of size at most $s(|\psi|)$, where $|\psi|$ stands for the length of ψ. ∎

To show the decidability of the satisfiability and finite satisfiability problems for S_p, it suffices to establish the small model property for S_p. To do this, we use a path label criterion to characterize whether a σ-structure satisfies a sentence of S_p. More specifically, given a structure G and a sentence ψ of S_p, we label each node of G with paths in ψ. The path label of G, $LB(G, \psi)$, is the collection of the labels of all the nodes in G. This path label has the following properties:

- for any σ-structure H, if $LB(H, \psi) = LB(G, \psi)$, then $H \models \psi$ iff $G \models \psi$; and
- there is a σ-structure H of size at most $2^{2^{2|\psi|}}$, such that $LB(H, \psi) = LB(G, \psi)$.

In the remainder of this section, we present the path label criterion and show that it has the properties described above.

5.2.1 A path label criterion

We first define the path labels, and then discuss their properties.

Path labels

Given a σ-structure G and a sentence ψ in S_p, we define a path label $LB(G, \psi)$ to characterize whether $G \models \psi$.

Let $G = (|G|, r^G, E^G)$ and $\psi = \bigwedge \Sigma \wedge \neg\varphi$. We use the following sets to denote the paths in ψ:

$$
\begin{aligned}
Paths_{\alpha}(\psi) &= \{pf(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\} \\
Paths_{\beta}(\psi) &= \{lt(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\} \\
Paths^{+}_{\gamma}(\psi) &= \{rt(\phi) \mid \phi \in \Sigma \cup \{\varphi\},\ \phi \text{ is a forward constraint}\} \\
Paths^{-}_{\gamma}(\psi) &= \{-rt(\phi) \mid \phi \in \Sigma \cup \{\varphi\},\ \phi \text{ is a backward constraint}\} \\
Paths_{(\beta,\gamma)}(\psi) &= Paths_{\beta}(\psi) \cup Paths^{+}_{\gamma}(\psi) \cup Paths^{-}_{\gamma}(\psi)
\end{aligned}
$$

Here the notation $-\rho$ denotes the pair $(-, \rho)$. We use this notation merely to distinguish the occurrence of a path as the right tail of a backward constraint as opposed to a forward constraint.

For each node a in $|G|$, we define a path label using paths in $Paths_\alpha(\psi)$ and $Paths_{(\beta,\gamma)}(\psi)$. This label consists of a pair of sets. The first component of the pair is the set of paths from r^G to a which are in $Paths_\alpha(\psi)$. That is,

$$lb_\alpha(a, G, \psi) = \{\rho \mid \rho \in Paths_\alpha(\psi),\ G \models \rho(r^G, a)\}.$$

The second is a collection of sets of paths in $Paths_{(\beta,\gamma)}(\psi)$. Each set consists of the paths between the node a and a node in $|G|$. More specifically, for each $b \in |G|$, let:

$$\begin{aligned}
lbs_\beta(a, b, G, \psi) &= \{\rho \mid \rho \in Paths_\beta(\psi),\ G \models \rho(a, b)\} \\
lbs_\gamma(a, b, G, \psi) &= \{\rho \mid \rho \in Paths^+_\gamma(\psi),\ G \models \rho(a, b)\} \\
&\quad \cup \{-\rho \mid -\rho \in Paths^-_\gamma(\psi),\ G \models \rho(b, a)\} \\
lbs_{(\beta,\gamma)}(a, b, G, \psi) &= lbs_\beta(a, b, G, \psi) \ \cup\ lbs_\gamma(a, b, G, \psi)
\end{aligned}$$

The second component of the label is defined by:

$$lb_{(\beta,\gamma)}(a, G, \psi) = \{lbs_{(\beta,\gamma)}(a, b, G, \psi) \mid b \in |G|\}$$

More precisely, we define the *label of node a in G w.r.t. ψ* by:

$$lb(a, G, \psi) = \begin{cases} (\emptyset,\ \emptyset) & \text{if } lb_\alpha(a, G, \psi) = \emptyset \\ (lb_\alpha(a, G, \psi),\ lb_{(\beta,\gamma)}(a, G, \psi)) & \text{otherwise} \end{cases}$$

The *label of G w.r.t. ψ* is defined by

$$LB(G, \psi) \ = \ \{lb(a, G, \psi) \mid a \in |G|\}.$$

Every label $l \in LB(G, \psi)$ is a pair of sets. We refer to the first component of l as $lb_\alpha(l)$, and the second as $lb_{(\beta,\gamma)}(l)$. In addition, we use the following notations:

$$\begin{aligned}
LB_\alpha(G, \psi) &= \{lb_\alpha(l) \mid l \in LB(G, \psi)\} \\
LB_{(\beta,\gamma)}(G, \psi) &= \{lb_{(\beta,\gamma)}(l) \mid l \in LB(G, \psi)\}
\end{aligned}$$

We now consider a special case of $LB(G, \psi)$. If ψ involves simple constraints only, i.e., $\Sigma \cup \{\varphi\}$ is a subset of P_s, then $Paths_\alpha(\psi) = \{\epsilon\}$. Thus we have:

$$LB(G,\psi) = \begin{cases} \{(\epsilon,\ lb_{(\beta,\gamma)}(r^G, G, \psi))\} & \text{if } |G| \text{ is a singleton set} \\ \{(\epsilon,\ lb_{(\beta,\gamma)}(r^G, G, \psi)),\ (\emptyset,\ \emptyset)\} & \text{otherwise} \end{cases}$$

In this case, the cardinality of $LB(G, \psi)$ is at most 2.

Properties of the path labels

The most important property of $LB(G, \psi)$ is that it characterizes whether $G \models \psi$.

Let G be a σ-structure and ψ a sentence, as described above. We say that $LB(G, \psi)$ *satisfies* ψ iff it satisfies the following conditions.

- For each $\phi \in \Sigma$, $LB(G, \psi)$ satisfies ϕ. That is, for any $l \in LB(G, \psi)$ and $s \in lb_{(\beta,\gamma)}(l)$, if $pf(\phi) \in lb_\alpha(l)$ and $lt(\phi) \in s$, then
  - $rt(\phi) \in s$ if ϕ is a forward constraint, and
  - $-rt(\phi) \in s$ if ϕ is a backward constraint.
- $LB(G, \psi)$ does not satisfy φ. That is, there exists $l \in LB(G, \psi)$ and $s \in lb_{(\beta,\gamma)}(l)$, such that $pf(\varphi) \in lb_\alpha(l)$, $lt(\varphi) \in s$, and
  - $rt(\varphi) \notin s$ if φ is a forward constraint, and
  - $-rt(\varphi) \notin s$ if φ is a backward constraint.

Lemma 5.2: For any σ-structure G and any sentence $\psi \in S_p$, $G \models \psi$ iff $LB(G, \psi)$ satisfies ψ. ∎

Proof: Let $\psi = \bigwedge \Sigma \wedge \neg\varphi$. It suffices to show that for each $\phi \in \Sigma \cup \{\varphi\}$, $G \models \phi$ iff $LB(G, \psi)$ satisfies ϕ. Without loss of generality, assume that all the constraints in $\Sigma \cup \{\varphi\}$ are forward constraints. The proof for the backward case is analogous.

(1) Assume $G \models \phi$, we want to show that $LB(G, \psi)$ satisfies ϕ.

Suppose, for *reductio*, that $LB(G,\psi)$ does not satisfy ϕ. That is, there exist $l \in LB(G,\psi)$ and $s \in lb_{(\beta,\gamma)}(l)$, such that $pf(\phi) \in lb_\alpha(l)$ and $lt(\phi) \in s$, but $rt(\phi) \notin s$. By the definition of the path labels, there are $a, b \in |G|$ such that $lb(a,G,\psi) = l$ and $lbs_{(\beta,\gamma)}(a,b,G,\psi) = s$. Hence,

$$G \models pf(\phi)(r^G, a) \wedge lt(\phi)(a,b) \wedge \neg rt(\phi)(a,b).$$

This contradicts the assumption.

(2) Conversely, assume $G \not\models \phi$. we want to show that $LB(G,\psi)$ does not satisfy ϕ.

Suppose, for *reductio*, that $LB(G,\psi)$ satisfies ϕ. That is, for every $l \in LB(G,\psi)$ and every $s \in lb_{(\beta,\gamma)}(l)$, if $pf(\phi) \in lb_\alpha(l)$ and $lt(\phi) \in s$, then $rt(\phi) \in s$. However, since $G \models \neg\phi$, there exist $a, b \in |G|$, such that

$$G \models pf(\phi)(r^G, a) \wedge lt(\phi)(a,b) \wedge \neg rt(\phi)(a,b).$$

Thus by the definition of the path labels, $pf(\phi) \in lb_\alpha(a,G,\psi)$, $lt(\phi) \in lbs_{(\beta,\gamma)}(a,b,G,\psi)$, but $rt(\phi) \notin lbs_{(\beta,\gamma)}(a,b,G,\psi)$. Let $l = lb(a,G,\psi)$ and $s = lbs_{(\beta,\gamma)}(a,b,G,\psi)$. Clearly, l and s contradict the assumption. ∎

From Lemma 5.2 follows immediately the corollary below.

Corollary 5.3: For all structures G, H, and any sentence $\psi \in S_p$, if $LB(G,\psi) = LB(H,\psi)$, then $G \models \psi$ iff $H \models \psi$. ∎

The size of a path label

We next examine the cardinality of $LB(G,\psi)$. We use $|S|$ to denote the cardinality of a set S.

Given a sentence $\psi \in S_p$, where $\psi = \bigwedge \Sigma \wedge \neg\varphi$, it is easy to verify that

$$\begin{aligned} |Paths_\alpha(\psi)| &\leq |\psi|, \\ |Paths_{(\beta,\gamma)}(\psi)| &\leq |\psi|. \end{aligned}$$

For every σ-structure G and every $l \in LB(G,\psi)$, $lb_\alpha(l)$ is a subset of $Paths_\alpha(\psi)$ and

$lb_{(\beta,\gamma)}(l)$ is a subset of the power set of $Paths_{(\beta,\gamma)}(\psi)$. Therefore,

$$|LB(G,\psi)| \leq 2^{|\psi| + 2^{|\psi|}}.$$

In particular, if ψ involves simple constraints only, then $|LB(G,\psi)| \leq 2$.

We define the *prefix length of* ψ, $s_\alpha(\psi)$, to be $|pf(\varphi)|$. Note that the prefixes of all the constraints in $\Sigma \cup \{\varphi\}$ have the same length.

5.2.2 The small model property

Next, we establish the small model property for S_p. Using the path label criterion described above, it suffices to show the following.

Proposition 5.4: For each σ-structure G and each sentence ψ in S_p, there is a σ-structure H, such that

1. the size of H is at most $2^{2^{2|\psi|}}$; and
2. $LB(H,\psi) = LB(G,\psi)$.

■

The proof of the proposition requires two lemmas and the following notation.

Definition 5.5: Let G be a σ-structure, m be a natural number and $a \in |G|$. The *m-neighborhood of a in G* is the structure $G(a) = (|G(a)|, r^{G(a)}, E^{G(a)})$, such that

- $|G(a)| = \{c \mid c \in |G|$, there is path ρ, $|\rho| \leq m$ and $G \models \rho(a,c) \vee \rho(c,a)\}$;
- $r^{G(a)} = a$; and
- for all $b, c \in |G(a)|$ and each $K \in E$, $G(a) \models K(b,c)$ iff $G \models K(b,c)$.

That is, $G(a)$ is the restriction of G to $|G(a)|$ with a as the new root. ■

Given a σ-structure G and a sentence ψ in S_p, the first lemma below proves the existence of a σ-structure G_α which has the following properties.

- $LB_\alpha(G_\alpha, \psi) = LB_\alpha(G, \psi)$. In addition, for each $l \in LB(G, \psi)$, there is a distinguished node $a_l \in |G_\alpha|$ such that $lb_\alpha(a_l, G_\alpha, \psi) = lb_\alpha(l)$.
- For each $a \in |G_\alpha|$, if $lb_\alpha(a, G_\alpha, \psi) \neq \emptyset$, then a does not have any outgoing edge. That is, for each $K \in E$ and $b \in |G_\alpha|$, $G_\alpha \models \neg K(a, b)$.

We shall proceed to construct the σ-structure H described in Proposition 5.4 such that G_α is the $s_\alpha(\psi)$-neighborhood of r^H in H. This ensures that $LB_\alpha(H, \psi) = LB_\alpha(G, \psi)$.

Lemma 5.5: For each σ-structure G and each sentence ψ in S_p, there exists a σ-structure $G_\alpha = (|G_\alpha|, r^{G_\alpha}, E^{G_\alpha})$, such that

1. the size of G_α is at most $|\psi| + 2^{|\psi| + 2|\psi|}$;
2. there is a subset L_α of $|G_\alpha|$, such that
   (a) there is a bijection $f : LB(G, \psi) \to L_\alpha$, such that for each $l \in LB(G, \psi)$,
      i. $lb_\alpha(l) = lb_\alpha(f(l), G_\alpha, \psi) = \{\rho \mid \rho \text{ is a path, } G_\alpha \models \rho(r^{G_\alpha}, f(l))\}$,
      ii. for each $K \in E$ and $b \in |G_\alpha|$, $G_\alpha \models \neg K(f(l), b)$;
   (b) for each $b \in |G_\alpha| \setminus L_\alpha$,
      i. $lb_\alpha(b, G_\alpha, \psi) = \emptyset$,
      ii. there is a unique path ρ such that $G_\alpha \models \rho(r^{G_\alpha}, b)$. In addition, $|\rho| < s_\alpha(\psi)$. ∎

Proof: Let $I_\alpha(\psi) = \bigcup_{\varrho \in Paths_\alpha(\psi)} \{\rho \mid \rho \prec_p \varrho\}$. Here $\rho \prec_p \varrho$ stands for that ρ is a proper prefix of ϱ, as defined in Chapter 2. We construct G_α using $LB(G, \psi)$ and $I_\alpha(\psi)$ as follows. For each $\rho \in I_\alpha(\psi)$, let a_ρ be a distinguished node, and for each $l \in LB(G, \psi)$, let a_l be a distinguished node. Let

- $L_\alpha = \{a_l \mid l \in LB(G, \psi)\}$;
- $|G_\alpha| = L_\alpha \cup \{a_\rho \mid \rho \in I_\alpha(\psi)\}$;
- $r^{G_\alpha} = \begin{cases} a_\epsilon & \text{if } s_\alpha(\psi) \geq 1 \\ a_{lb(r^G, G, \psi)} & \text{otherwise;} \end{cases}$

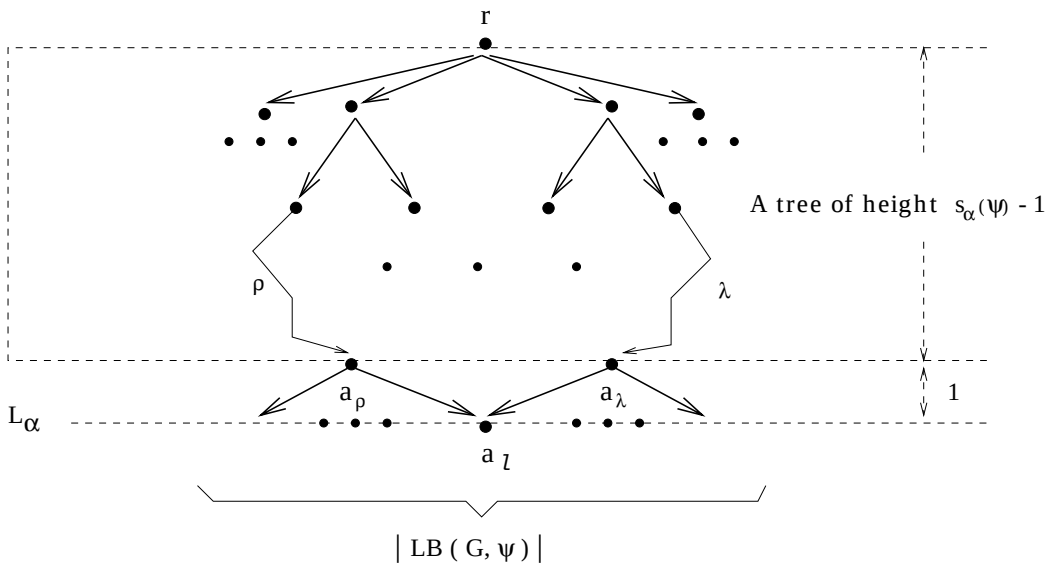

Figure 5.1: The structure G_α in Lemma 5.5

- for all $a, b \in |G_\alpha|$ and $K \in E$, $G_\alpha \models K(a,b)$ iff there exists $\rho \in I_\alpha(\psi)$, such that $a = a_\rho$ (i.e., $a \notin L_\alpha$), and one of the following conditions is satisfied:
  - there exists $\varrho \in I_\alpha(\psi)$, such that $b = a_\varrho$ (i.e., $b \notin L_\alpha$), and $\varrho = \rho \cdot K$; or
  - there exists $l \in LB(G,\psi)$, such that $b = a_l$ (i.e., $b \in L_\alpha$), and there exists $\varrho \in lb_\alpha(l)$, such that $\varrho = \rho \cdot K$.

The structure G_α is basically a rooted acyclic directed graph (see Figure 5.1). It has the following properties.

- The restriction of G_α to $\{a_\rho \mid \rho \in I_\alpha(\psi)\}$ is a tree of height $s_\alpha(\psi) - 1$. For each node a_ρ in the tree, there is a single path ρ from the root r^{G_α} to a_ρ.
- At level $s_\alpha(\psi)$, there are $|LB(G,\psi)|$ many nodes. Each of these nodes is uniquely marked with a label $l \in LB(G,\psi)$. In addition, it does not have any outgoing edges, and all its incoming edges are from leaves of the tree mentioned above.

We now verify that G_α indeed meets all the requirements of the lemma.

(1) *The size of* G_α.

Let $size(A)$ denote the size of a structure A. Since $|L_\alpha| = |LB(G,\psi)| \leq 2^{|\psi|+2^{|\psi|}}$ and $|I_\alpha(\psi)| \leq |\psi|$, $size(G_\alpha)$ is at most

$$|\psi| + 2^{|\psi|+2^{|\psi|}}.$$

In particular, when $s_\alpha(\psi) = 0$, $|LB(G,\psi)| \leq 2$ and $size(G_\alpha)$ is at most 2.

(2) *The properties of* L_α.

The bijection f from $LB(G,\psi)$ to L_α can be defined by: $l \mapsto a_l$. To verify the other properties of L_α, first observe the following simple fact.

Claim: For every $\varrho \in I_\alpha(\psi)$, $\{\rho \mid \rho \text{ is a path, } G_\alpha \models \rho(r^{G_\alpha}, a_\varrho)\} = \{\varrho\}$.

This claim can be verified by a straightforward induction on $|\varrho|$. From this claim and the construction of G_α, the second statement of the lemma follows. ∎

The next lemma deals with $LB_{(\beta,\gamma)}(G,\psi)$. Given a label l in $LB(G,\psi)$, it constructs a σ-structure $G_l = (|G_l|, r^{G_l}, E^{G_l})$ such that

$$lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l).$$

We shall construct the structure H described in Proposition 5.4 in such a way that for each l in $LB(G,\psi)$, G_l is part of H, and moreover,

$$lb_{(\beta,\gamma)}(r^{G_l}, H, \psi) = lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi).$$

Lemma 5.6: Let G be a σ-structure and ψ a sentence in S_p. For each label l in $LB(G,\psi)$, there is a σ-structure G_l, such that

1. the size of G_l is at most $2^{|\psi|}$; and
2. $lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l)$.

∎

Proof: We give a filtration argument. To do this, we need the following notations.

First, we define the following sets:

$$\begin{aligned} I^+(\psi) &= \bigcup_{\varrho \in Paths_\beta(\psi) \cup Paths^+_\gamma(\psi)} \{\rho \mid \rho \preceq_p \varrho\} \\ I^-(\psi) &= \bigcup_{-\varrho \in Paths^-_\gamma(\psi)} \{-\rho \mid \rho \preceq_s \varrho\} \\ I(\psi) &= I^+(\psi) \cup I^-(\psi) \end{aligned}$$

Here $\rho \preceq_p \varrho$ denotes that ρ is a prefix of ϱ, and $\rho \preceq_s \varrho$ denotes that ρ is a suffix of ϱ, as described in Chapter 2. It is easy to verify that $|I(\psi)| \leq |\psi|$.

Second, by $l \in LB(G, \psi)$, there exists $a \in |G|$ such that

$$lb(a, G, \psi) = l.$$

Using a, we define a mapping g from $|G|$ to the power set of $I(\psi)$, such that for each $b \in |G|$,

$$g(b) \mapsto \{\rho \mid \rho \in I^+(\psi),\, G \models \rho(a,\, b)\} \cup \{-\rho \mid -\rho \in I^-(\psi),\, G \models \rho(b,\, a)\}.$$

Using the mapping g, we define an equivalence relation $\sim$ on $|G|$ such that

$$b \sim b' \quad \text{iff} \quad g(b) = g(b').$$

Let $[b]$ denote the equivalence class of b with respect to $\sim$. We proceed to construct a σ-structure $G_l = (|G_l|,\, r^{G_l},\, E^{G_l})$ whose nodes are these equivalence classes:

- $|G_l| = \{[b] \mid b \in |G|\}$;
- $r^{G_l} = [a]$;
- for all $o_1, o_2 \in |G_l|$ and $K \in E$, $G_l \models K(o_1, o_2)$ iff there exist $b_1, b_2 \in |G|$, such that $[b_1] = o_1$, $[b_2] = o_2$, and $G \models K(b_1,\, b_2)$.

We next show that G_l is indeed the structure desired.

(1) *The size of* G_l.

For each $b \in |G|$, $g(b) \subseteq I(\psi)$. Since $|I(\psi)| \leq |\psi|$, the size of G_l is at most $2^{|\psi|}$.

(2) $lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l)$.

It suffices to show the following claim.

Claim 1: For each $b \in |G|$, $lbs_{(\beta,\gamma)}(r^{G_l}, [b], G_l, \psi) = lbs_{(\beta,\gamma)}(a, b, G, \psi)$.

For if Claim 1 holds, then

$$\begin{aligned} lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) &= \{lbs_{(\beta,\gamma)}(r^{G_l}, c, G_l, \psi) \mid c \in |G_l|\} \\ &= \{lbs_{(\beta,\gamma)}(r^{G_l}, [b], G_l, \psi) \mid b \in |G|\} \\ &= \{lbs_{(\beta,\gamma)}(a, b, G, \psi) \mid b \in |G|\} \\ &= lb_{(\beta,\gamma)}(a, G, \psi). \\ &= lb_{(\beta,\gamma)}(l). \end{aligned}$$

To verify Claim 1, it suffices to show the following.

Claim 2: For every $b \in |G|$ and $\rho \in I^+(\psi)$, $G \models \rho(a, b)$ iff $G_l \models \rho(r^{G_l}, [b])$.

Claim 3: For every $b \in |G|$ and $-\rho \in I^-(\psi)$, $G \models \rho(b, a)$ iff $G_l \models \rho([b], r^{G_l})$.

For if these claims hold, then from $Paths_{(\beta,\gamma)}(\psi) \subseteq I(\psi)$ and the definition of $lbs_{(\beta,\gamma)}$ follows Claim 1.

We next show Claim 2 by induction on $|\rho|$. Similarly, Claim 3 can be verified.

Base case: $|\rho| = 0$. That is, $\rho = \epsilon$. By the definition of g, it is straightforward to verify that $G_l \models \epsilon(r^{G_l}, [b])$ iff $g(b) = g(a)$ iff $\epsilon \in g(b)$ iff $b = a$ iff $G \models \epsilon(a, b)$. Therefore, Claim 2 holds in this case.

Inductive step: Assume the claim for $|\rho| = m$.

We next show that the claim holds for $|\rho| = m + 1$. That is, ρ is of the form $\varrho \cdot K$, where $\varrho \in I^+(\psi)$, $|\varrho| = m$ and $K \in E$.

First, suppose that $G \models \rho(a,\, b)$. Then there exists $c \in |G|$, such that

$$G \models \varrho(a,\, c) \wedge K(c,\, b).$$

By the induction hypothesis,

$$G_l \models \varrho(r^{G_l},\, [c]).$$

Moreover, by $G \models K(c, b)$ and the definition of G_l, we have

$$G_l \models K([c],\, [b]).$$

Therefore, $G_l \models \rho(r^{G_l},\, [b])$.

Conversely, assume that $G_l \models \rho(r^{G_l},\, [b])$. Then there exists $o \in |G_l|$, such that

$$G_l \models \varrho(r^{G_l},\, o) \wedge K(o,\, [b]).$$

By the definition of G_l and $G_l \models K(o,\, [b])$, there exist $o_1, b_1 \in |G|$, such that $[o_1] = o$, $[b_1] = [b]$, and

$$G \models K(o_1,\, b_1).$$

In addition, since $G_l \models \varrho(r^{G_l},\, o)$ and $[o_1] = o$, by the induction hypothesis, we have that

$$G \models \varrho(a,\, o_1).$$

Therefore, $G \models \rho(a,\, b_1)$. That is, $\rho \in g(b_1)$. By $[b_1] = [b]$, we have that $g(b_1) = g(b)$. Therefore, $\rho \in g(b)$. Hence $G \models \rho(a,\, b)$.

This completes the proof of Lemma 5.6. ∎

Finally, we prove Proposition 5.4. As mentioned earlier, given a σ-structure G and a sentence ψ in S_p, we define the structure H described in Proposition 5.4 such that

- the structure G_α in Lemma 5.5 is the $s_\alpha(\psi)$-neighborhood of r^H in H;
- for each $l \in LB(G, \psi)$, G_l in Lemma 5.6 is part of H such that
  - $r^{G_l} = f(l)$, where f is the function specified in Lemma 5.5,

- $lb_{(\beta,\gamma)}(r^{G_l}, H, \psi) = lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l)$, and
- $lb_{\alpha}(r^{G_l}, H, \psi) = lb_{\alpha}(l)$.

Note that the proof below uses the restriction on prefixes described in Definition 5.1.

Proof of Proposition 5.4: Given a σ-structure G and a sentence ψ in S_p, let G_α be the structure specified in Lemma 5.5, and for each $l \in LB(G, \psi)$, let G_l be the structure specified in Lemma 5.6. Without loss of generality, assume that $|G_l| \cap |G_\alpha| = \emptyset$ and $|G_l| \cap |G_{l'}| = \emptyset$ if $l \neq l'$. We build σ-structure $H = (|H|, r^H, E^H)$, as follows.

- $|H| = |G_\alpha| \cup \bigcup_{l \in LB(G,\psi)} (|G_l| \setminus \{r^{G_l}\})$;
- $r^H = r^{G_\alpha}$;
- For all $a, b \in |H|$ and each $K \in E$, $H \models K(a, b)$ iff one of the following conditions is satisfied:
  - $a, b \in |G_\alpha|$ and $G_\alpha \models K(a, b)$;
  - For some $l \in LB(G, \psi)$, $a, b \in |G_l|$ and $G_l \models K(a, b)$;
  - Let L_α be the subset of $|G_\alpha|$ and f the function specified in Lemma 5.5. For some $l \in LB(G, \psi)$,
    - $a = f(l)$, $b \in |G_l|$ and $G_l \models K(r^{G_l}, b)$; or
    - $b = f(l)$, $a \in |G_l|$ and $G_l \models K(a, r^{G_l})$; or
    - $a = b = f(l)$ and $G_l \models K(r^{G_l}, r^{G_l})$.

Intuitively, H is built from G_α and G_l's by identifying $f(l)$ with r^{G_l} for each $l \in LB(G, \psi)$. See Figure 5.2 for the structure H.

We now show that H is indeed the structure desired.

(1) *The size of* H.

Obviously,

$$size(H) = size(G_\alpha) + \sum_{l \in LB(G,\psi)} size(G_l) - |LB(G, \psi)|.$$

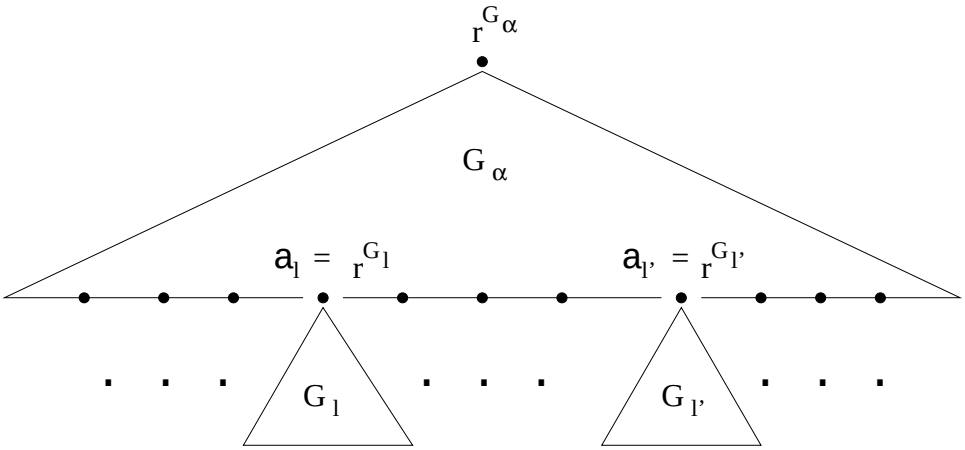

Figure 5.2: The structure H in Proposition 5.4

By Lemmas 5.5 and 5.6, the size of H is at most

$$|\psi| + 2^{|\psi| + 2^{|\psi|}}\ 2^{|\psi|},$$

which is no larger than $2^{2^{2|\psi|}}$.

Note that when $s_\alpha(\psi) = 0$, the size of H is at most $2^{|\psi|}$.

(2) $LB(H, \psi) = LB(G, \psi)$.

It suffices to show the claim below.

Claim: Let L_α be the set and f the function specified in Lemma 5.5. They have the following properties.

1. For each $a \in |H| \setminus L_\alpha$, $lb(a, H, \psi) = (\emptyset,\ \emptyset)$.
2. For each $l \in LB(G, \psi)$, $lb(f(l), H, \psi) = l$.

For if the claim holds, then $LB(G, \psi) \subseteq LB(H, \psi)$. In addition, by Lemma 5.5, f is a bijection between $LB(G, \psi)$ and L_α. Therefore, $LB(H, \psi) = LB(G, \psi)$.

To show the claim, first observe the following simple facts about H, which are immediate from the definition of H.

Fact 1: For any $l \in LB(G, \psi)$ and $a, b \in |H|$, if $b \in |G_l|$ and $a \notin |G_l|$, then for each path ρ,

- $H \models \rho(a,b)$ iff there are λ and ϱ, such that $\rho = \lambda \cdot \varrho$ and $H \models \lambda(a, f(l)) \wedge \varrho(f(l), b)$;
- $H \models \rho(b,a)$ iff $a = f(l)$ and $G_l \models \rho(b, r^{G_l})$.

Fact 2: For any $l \in LB(G,\psi)$ and $a \in |H|$, if there is path ρ such that $H \models \rho(f(l), a)$, then either $a \in |G_l|$ or $a = f(l)$.

Fact 3: For each $l \in LB(G,\psi)$, for any path ρ and node $a \in |G_l| \cap |H|$,

$$\begin{aligned} H \models \rho(f(l), a) \quad &\text{iff} \quad G_l \models \rho(r^{G_l}, a), \\ H \models \rho(a, f(l)) \quad &\text{iff} \quad G_l \models \rho(a, r^{G_l}), \\ H \models \rho(f(l), f(l)) \quad &\text{iff} \quad G_l \models \rho(r^{G_l}, r^{G_l}). \end{aligned}$$

Using the facts above, we examine the following cases.

Case 1: $a \in |G_\alpha| \setminus L_\alpha$. By Facts 1 and 2, all the paths from r^H to node a are in G_α. By Lemma 5.5, there is only one path from r^H to a, and the length of the path is less than $S_\alpha(\psi)$. By the definition of the path labels and the restriction on prefixes described in Definition 5.1, we have

$$lb(a, H, \psi) = (\emptyset,\ \emptyset).$$

Case 2: *For some* $l \in LB(G,\psi)$, $a \in |G_l| \setminus \{r^{G_l}\}$. By Fact 1, if there is path ρ from r^H to node a, then there must be paths λ and ϱ, such that $\rho = \lambda \cdot \varrho$ and $H \models \lambda(a, f(l)) \wedge \varrho(f(l), b)$. In addition, since $a \neq f(l)$, we have $\varrho \neq \epsilon$. By Lemma 5.5, $|\lambda| = s_\alpha(\psi)$. Hence $s_\alpha(\psi) < |\rho|$. Therefore, by the definition of the path labels and the restriction on prefixes described in Definition 5.1, we have

$$lb(a, H, \psi) = (\emptyset,\ \emptyset).$$

Case 3: $a \in L_\alpha$. That is, $a = f(l)$ for some $l \in LB(G,\psi)$. By Lemma 5.5 and Fact 1, $lb_\alpha(a, H, \psi) = lb_\alpha(l)$. By Facts 2, 3 and Lemma 5.6, we have

$$lb_{(\beta,\gamma)}(a, H, \psi) = lb_{(\beta,\gamma)}(l).$$

Hence $lb(a, H, \psi) = l$.

Therefore, the claim holds.

This completes the proof of Proposition 5.4 ∎

5.3 Decidability of the implication problems for P_β

In this section, we establish the decidability of the implication problems for P_β, which is the sublanguage of P_c defined in Section 5.1.2.

Theorem 5.7: In the context of $\mathcal{SM}$, the implication and finite implication problems for P_β are decidable. ∎

In the same way as in the proof of Theorem 5.1, we show Theorem 5.7 by establishing the small model property for the following set of sentences:

$$S(P_\beta) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \varphi \in P_\beta,\ \Sigma \subset P_\beta,\ \Sigma \text{ is finite}\}.$$

To do this, we give a filtration argument. Given a satisfiable sentence ψ in $S(P_\beta)$, we find the set of paths in ψ and use a path labeling mechanism similar to the one employed in the proof of Theorem 5.1. More specifically, let G be a model of ψ. We use the paths in ψ to label each node of G, and therefore, obtain the label of G with respect to ψ. The cardinality of this label is determined only by $|\psi|$, the length of ψ. We then construct a σ-structure H, such that H and G have the same label with respect to ψ, and moreover, $H \models \psi$. In addition, each node of H has a unique path label. The size of H is, therefore, bounded by the cardinality of the label of G with respect to ψ, which is at most $2^{|\psi|}$. Thus the small model property is established.

We first define the path labels, called *relative path labels*. Using the path labels, we then establish the small model property for $S(P_\beta)$.

5.3.1 Relative path label

Given a satisfiable sentence ψ of $S(P_\beta)$, where $\psi = \bigwedge \Sigma \wedge \neg\varphi$, we use the following sets to denote paths in ψ:

$$\begin{aligned} Paths_{(\alpha,\beta)}(\psi) &= \{pf(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\} \ \cup\ \{lt(\phi) \mid \phi \in \Sigma \cup \{\varphi\},\ \phi \in P_s\} \\ I_{(\alpha,\beta)}(\psi) &= \bigcup_{\varrho \in Paths_{(\alpha,\beta)}(\psi)} \{\rho \mid \rho \preceq_p \varrho\} \end{aligned}$$

$$I(\varphi) \quad = \quad \begin{cases} \{\rho \mid \rho \preceq_p rt(\varphi)\} & \text{if } \varphi \text{ is a forward constraint} \\ \{\rho \mid \rho \preceq_s rt(\varphi)\} & \text{if } \varphi \text{ is a backward constraint} \end{cases}$$

Here $\rho \preceq_p \varrho$ ($\rho \preceq_s \varrho$) means that ρ is a prefix (suffix) of ϱ, as defined in Chapter 2.

Let G be a model of ψ, $G = (|G|, r^G, E^G)$, and (a, b) be a pair of nodes in $|G|$ such that

$$G \models pf(\varphi)(r, a) \wedge lt(\varphi)(a, b) \wedge \neg rt(\varphi)(a, b)$$

if φ is a forward constraint, and

$$G \models pf(\varphi)(r, a) \wedge lt(\varphi)(a, b) \wedge \neg rt(\varphi)(b, a)$$

if φ is a backward constraint. This pair is referred to as a *witness of* $\neg\varphi$ *in* G.

For each $c \in |G|$, we label c with a pair. The first component of the pair is

$$ls_{(\alpha,\beta)}(c, G, \psi) = \{\rho \mid \rho \in I_{(\alpha,\beta)}(\psi),\ G \models \rho(r^G, c)\}.$$

The second component is defined to be:

$$ls_\varphi(c, a, G, \psi) \quad = \quad \begin{cases} \{\rho \mid \rho \in I(\varphi),\ G \models \rho(a, c)\} & \text{if } \varphi \text{ is a forward constraint} \\ \{\rho \mid \rho \in I(\varphi),\ G \models \rho(c, a)\} & \text{if } \varphi \text{ is a backward constraint} \end{cases}$$

The *path label of node* c *in* G *relative to* ψ *and* a is defined to be:

$$ls(c, G, \psi, a) = (ls_{(\alpha,\beta)}(c, G, \psi),\ ls_\varphi(c, a, G, \psi))$$

The *path label of* G *relative to* ψ *and* a is defined to be:

$$LS(G, \psi, a) = \{ls(c, G, \psi, a) \mid c \in |G|\}$$

Note that for each $c \in |G|$,

- $\epsilon \in ls_{(\alpha,\beta)}(c, G, \psi)$ iff $c = r^G$, and
- $\epsilon \in ls_\varphi(c, a, G, \psi)$ iff $c = a$.

We next examine the size of a relative path label.

Given a satisfiable sentence ψ of $S(P_\beta)$, where $\psi = \bigwedge \Sigma \wedge \neg\varphi$, let G be a model of ψ and (a, b) a witness of $\neg\varphi$ of G. Note that for each $c \in |G|$,

$$\begin{aligned} ls_{(\alpha,\beta)}(c, G, \psi) &\subseteq I_{(\alpha,\beta)}(\psi), \\ ls_{\varphi}(c, a, G, \psi) &\subseteq I(\varphi). \end{aligned}$$

In addition, it is easy to verify that

$$|I_{(\alpha,\beta)}(\psi)| + |I(\varphi)| \leq |\psi|.$$

Hence

$$|LS(G, \psi, a)| \leq 2^{|\psi|}.$$

The notion of relative path labels differs from the one described in Section 5.2.1 in the following aspects. First, relative path labels are defined for models of satisfiable sentences in $S(P_\beta)$, rather than for arbitrary σ-structures. Second, the relative path label of a node in a structure involves only the paths between the node and two fixed nodes of the structure, whereas the one given in Section 5.2.1 contains paths related to all the nodes in the structure. As a result, a relative path label has a much smaller cardinality. Third, a relative path label does not characterize whether a σ-structure is a model of a sentence in $S(P_\beta)$, but based on it we are able to form a filtration argument to establish the small model property for $S(P_\beta)$.

5.3.2 The small model property

Based on relative path labels we establish the following proposition, from which follows Theorem 5.7.

Proposition 5.8: Every satisfiable sentence ψ of $S(P_\beta)$ has a model of size at most $2^{|\psi|}$.

■

Proof: Let ψ be a satisfiable sentence in $S(P_\beta)$, where $\psi = \bigwedge \Sigma \wedge \neg\varphi$. Since ψ is satisfiable, there is a σ-structure $G = (|G|, r^G, E^G)$ such that $G \models \psi$. It follows that there

exist $a, b \in |G|$ such that (a, b) is a witness of $\neg\varphi$ in G. That is,

$$G \models pf(\varphi)(r, a) \wedge lt(\varphi)(a, b) \wedge \neg rt(\varphi)(a, b)$$

if φ is a forward constraint, and

$$G \models pf(\varphi)(r, a) \wedge lt(\varphi)(a, b) \wedge \neg rt(\varphi)(b, a)$$

if φ is a backward constraint.

Consider $LS(G, \psi, a)$. As in the proof of Lemma 5.6, we define an equivalence relation $\sim$ on $|G|$ by:

$$b \sim b' \quad \text{iff} \quad ls(b, G, \psi, a) = ls(b', G, \psi, a).$$

We denote the equivalence class of b with respect to $\sim$ as $[b]$. By taking these equivalence classes as nodes, we proceed to construct a σ-structure $H = (|H|, r^H, E^H)$ as follows:

- $|H| = \{[b] \mid b \in |G|\}$;
- $r^H = [r^G]$;
- for each $K \in E$ and $o_1, o_2 \in |H|$, $H \models K(o_1, o_2)$ iff there are $b_1, b_2 \in |G|$ such that $[b_1] = o_1$, $[b_2] = o_2$, and $G \models K(b_1, b_2)$.

We next show that $H \models \psi$, and moreover, the size of H is at most $2^{|\psi|}$.

(1) *The size of H.*

Obviously, the size of $|H|$ is at most $|LS(G, \psi, a)|$. Therefore, $size(H)$ is at most $2^{|\psi|}$.

(2) $H \models \psi$.

It suffices to show following claims.

Claim 1: For any path ρ and all $c, d \in |G|$, if $G \models \rho(c, d)$, then $H \models \rho([c], [d])$.

Claim 2: For each $c \in |G|$, $ls(c, G, \psi, a) = ls([c], H, \psi, [a])$.

Using these claims, we show $H \models \psi$ as follows. The proofs of these claims will be given shortly.

We first show that $H \models \Sigma$. Suppose, for *reductio*, that there exists $\phi \in \Sigma$ such that $H \models \neg\phi$. Without loss of generality, assume that ϕ is a forward constraint (the argument for the backward case is analogous). Then there exist $c, d \in |H|$, such that

$$H \models pf(\phi)(r^H, c) \wedge lt(\phi)(c, d) \wedge \neg rt(\phi)(c, d).$$

We have two cases to consider.

Case 1: ϕ is a simple constraint. That is, $pf(\phi) = \epsilon$ and $c = r^H$.

In this case, the assumption is equivalent to

$$lt(\phi) \in ls_{(\alpha,\beta)}(d, H, \psi) \quad \text{and} \quad H \models \neg rt(\phi)(r^H, d).$$

By the definition of H, there exists $d_1 \in |G|$, such that $[d_1] = d$. By Claim 2,

$$ls_{(\alpha,\beta)}(d_1, G, \psi) = ls_{(\alpha,\beta)}(d, H, \psi).$$

Thus $lt(\phi) \in ls_{(\alpha,\beta)}(d_1, G, \psi)$. That is, $G \models lt(\phi)(r^G, d_1)$. By $G \models \phi$, $G \models rt(\phi)(r^G, d_1)$. By Claim 1, we have $H \models rt(\phi)(r^H, d)$. This contradicts the assumption.

Case 2: ϕ is a β-restricted constraint, i.e., $|lt(\phi)| \leq 1$.

If $|lt(\phi)| = 0$, then $c = d$. Thus by the assumption,

$$pf(\phi) \in ls_{(\alpha,\beta)}(c, H, \psi) \quad \text{and} \quad H \models \neg rt(\phi)(c, c).$$

By the definition of H, there exists $c_1 \in |G|$, such that $[c_1] = c$. By Claim 2,

$$ls_{(\alpha,\beta)}(c_1, G, \psi) = ls_{(\alpha,\beta)}(c, H, \psi).$$

Thus $pf(\phi) \in ls_{(\alpha,\beta)}(c_1, G, \psi)$. That is, $G \models pf(\phi)(r^G, c_1)$. By $G \models \phi$, $G \models rt(\phi)(c_1, c_1)$. Thus by Claim 1, we have $H \models rt(\phi)(c, c)$. This contradicts the assumption.

If $|lt(\phi)| = 1$, then $lt(\phi) = K$ for some $K \in E$. By the assumption, we have

$$pf(\phi) \in ls_{(\alpha,\beta)}(c, H, \psi) \quad \text{and} \quad H \models K(c, d) \wedge \neg rt(\phi)(c, d).$$

By the definition of H, there exist nodes $c_1, d_1 \in |G|$, such that $[c_1] = c$, $[d_1] = d$ and $G \models K(c_1, d_1)$. By Claim 2, we have that

$$ls_{(\alpha,\beta)}(c_1, G, \psi) = ls_{(\alpha,\beta)}(c, H, \psi).$$

As a result, we have $G \models pf(\phi)(r^G, c_1)$. Thus $G \models pf(\phi)(r^G, c_1) \wedge K(c_1, d_1)$. By $G \models \phi$, $G \models rt(\phi)(c_1, d_1)$. Thus by Claim 1, we have that $H \models rt(\phi)(c, d)$. Again, this contradicts the assumption.

Therefore, $H \models \Sigma$.

We next show that $H \models \neg\varphi$. Since (a, b) is a witness of $\neg\varphi$ in G,

$$G \models pf(\varphi)(r^G, a) \wedge lt(\varphi)(a, b).$$

By Claim 1,

$$H \models pf(\varphi)(r^H, [a]) \wedge lt(\varphi)([a], [b]).$$

By Claim 2, we have that $ls_\varphi(b, a, G, \psi) = ls_\varphi([b], [a], H, \psi)$. As a result, when φ is a forward constraint, by $G \models \neg rt(\varphi)(a, b)$, we have that

$$H \models \neg rt(\varphi)([a], [b]);$$

and when φ is a backward constraint, by $G \models \neg rt(\varphi)(b, a)$, we have that

$$H \models \neg rt(\varphi)([b], [a]).$$

Therefore, $H \models \neg\varphi$.

We now show Claim 1 by induction on $|\rho|$.

Base case: If $|\rho| = 0$, then $c = d$. Hence clearly $[c] = [d]$. That is, $H \models \epsilon([c], [d])$.

Inductive step: Assume the claim for $|\rho| = m$.

We now consider ρ with $|\rho| = m + 1$. By $|\rho| = m + 1$, there are path ϱ and label $K \in E$, such that $\rho = \varrho \cdot K$ and $|\varrho| = m$. By $G \models \rho(c, d)$, there exists a node $c' \in |G|$, such that

$$G \models \varrho(c, c') \wedge K(c', d).$$

By the induction hypothesis, $H \models \varrho([c], [c'])$. Furthermore, by the definition of H, we have $H \models K([c'], [d])$. Hence $H \models \rho([c], [d])$.

Finally, we show Claim 2 by *reductio*.

Suppose that there exists $c \in |G|$, such that

$$ls(c, G, \psi, a) \neq ls([c], H, \psi, [a]).$$

Then we examine the following three cases.

Case 1: $ls_{(\alpha,\beta)}(c, G, \psi) \neq ls_{(\alpha,\beta)}([c], H, \psi)$.

To see this assumption leads to a contradiction, it suffices to show the claim below.

Claim 3: For each $\rho \in I_{(\alpha,\beta)}(\psi)$ and $c \in |G|$, $\rho \in ls_{(\alpha,\beta)}(c, G, \psi)$ iff $\rho \in ls_{(\alpha,\beta)}([c], H, \psi)$.

We show this claim by induction on $|\rho|$.

Base case: $|\rho| = 0$. That is, $\rho = \epsilon$. It is easy to see that

$$\epsilon \in ls_{(\alpha,\beta)}(c, G, \psi) \quad \text{iff} \quad c = r^G,$$
$$\epsilon \in ls_{(\alpha,\beta)}([c], H, \psi) \quad \text{iff} \quad [c] = [r^G].$$

Thus by the definition of f, we have that $\epsilon \in ls_{(\alpha,\beta)}(c, G, \psi)$ iff $\epsilon \in ls_{(\alpha,\beta)}([c], H, \psi)$.

Inductive step: Assume the claim for $|\rho| = m$. We next consider the claim for $|\rho| = m + 1$.

Suppose $\rho \in ls_{(\alpha,\beta)}(c, G, \psi)$. That is, $G \models \rho(r^G, c)$. Then by Claim 1, $H \models \rho(r^H, [c])$. That is, $\rho \in ls_{(\alpha,\beta)}([c], H, \psi)$.

Conversely, assume that $\rho \in ls_{(\alpha,\beta)}([c], H, \psi)$. Then there exist $d \in |H|$, $K \in E$ and path $\varrho \in I_{(\alpha,\beta)}(\psi)$, such that $\rho = \varrho \cdot K$, $|\varrho| = m$ and

$$H \models \varrho(r^H, d) \wedge K(d, [c]).$$

Because $H \models K(d, [c])$, by the definition of H, there exist nodes $c_1, d_1 \in |G|$ such that $[c_1] = [c]$, $[d_1] = d$ and $G \models K(d_1, c_1)$. Moreover, by the induction hypothesis, we have $\varrho \in ls_{(\alpha,\beta)}(d_1, G, \psi)$. That is, $G \models \varrho(r^G, d_1)$. Hence

$$\rho \in ls_{(\alpha,\beta)}(c_1, G, \psi).$$

By $[c_1] = [c]$, we have that $ls_{(\alpha,\beta)}(c, G, \psi) = ls_{(\alpha,\beta)}(c_1, G, \psi)$. Thus $\rho \in ls_{(\alpha,\beta)}(c, G, \psi)$.

Therefore, Claim 3 holds. As a result, the assumption in Case 1 leads to a contradiction.

Case 2: $ls_\varphi(c, a, G, \psi) \neq ls_\varphi([c], [a], G, \psi)$.

As for Case 1, it suffices to prove the following claim.

Claim 4: For each $\rho \in I(\varphi)$ and each $c \in |G|$, $\rho \in ls_\varphi(c, a, G, \psi)$ iff $\rho \in ls_\varphi([c], [a], H, \psi)$.

The proof of Claims 4 is similar to that of Claim 1, by induction on $|\rho|$. Here we assume that φ is a backward constraint. The proof for the forward case is analogous.

Base case: $|\rho| = 0$. That is, $\rho = \epsilon$. It is easy to see that

$$\epsilon \in ls_\varphi(c, a, G, \psi) \quad \text{iff} \quad c = a,$$
$$\epsilon \in ls_\varphi([c], [a], H, \psi) \quad \text{iff} \quad [c] = [a].$$

By the definition of f, we have that $\epsilon \in ls_\varphi(c, a, G, \psi)$ iff $\epsilon \in ls_\varphi([c], [a], H, \psi)$.

Inductive step: Assume the claim for $|\rho| = m$. We next consider the claim for $|\rho| = m + 1$.

Assume $\rho \in ls_\varphi(c, a, G, \psi)$. That is, $G \models \rho(c, a)$. Then by Claim 1, $H \models \rho([c], [a])$. That is, $\rho \in ls_\varphi([c], [a], H, \psi)$.

Conversely, assume that $\rho \in ls_\varphi([c], [a], H, \psi)$. Then there exist $d \in |H|$, $\varrho \in I(\varphi)$ and $K \in E$, such that $\rho = K \cdot \varrho$, $|\varrho| = m$ and

$$H \models K([c],\, d) \wedge \varrho(d,\, [a]).$$

Since $H \models K([c],\, d)$, by the definition of H, there are $c_1, d_1 \in |G|$ such that $[c_1] = [c]$, $[d_1] = d$ and $G \models K(c_1, d_1)$. By the induction hypothesis, we have that $\varrho \in ls_\varphi(d_1, G, \psi)$. That is, $G \models \varrho(d_1, a)$. Hence

$$\rho \in ls_\varphi(c_1, a, G, \psi).$$

By $[c_1] = [c]$, we have that $ls_\varphi(c, a, G, \psi) = ls_\varphi(c_1, a, G, \psi)$. Thus $\rho \in ls_\varphi(c, a, G, \psi)$.

Therefore, Claim 4 holds. Hence the assumption in Case 2 also leads to a contradiction.

This completes the proof of Proposition 5.8. ∎

5.4 Decidability of the extended implication for P_β

In this section, we prove the decidability of the extended implication problems for P_β.

Theorem 5.9: In the context of $\mathcal{SM}$, the extended implication and finite implication problems for P_β are decidable. ∎

We prove the theorem by reduction to the implication problems for P_β, whose decidability is established by Theorem 5.7.

Let $Paths$ denote the set of all paths, and let

$$S_e(P_\beta) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \Sigma \cup \{\varphi\} \text{ is a prefix extended subset of } P_\beta\}.$$

Recall the set $S(P_\beta)$ defined in the proof of Theorem 5.7. We define the *prefix extension function from* $S(P_\beta)$ *to* $S_e(P_\beta)$ to be the mapping

$$f : S(P_\beta) \times Paths \to S_e(P_\beta),$$

such that

$$f(\bigwedge \Sigma \wedge \neg\varphi,\ \alpha) \mapsto \bigwedge_{\phi\in\Sigma} \delta(\phi,\alpha) \wedge \neg\delta(\varphi,\alpha).$$

To prove Theorem 5.9, it suffices to show the proposition below.

Proposition 5.10: Let ψ be a sentence in $S(P_\beta)$, α a path, and f the prefix extension function from $S(P_\beta)$ to $S_e(P_\beta)$. Then

1. ψ is satisfiable iff $f(\psi,\alpha)$ is satisfiable;
2. ψ is finitely satisfiable iff $f(\psi,\alpha)$ is finitely satisfiable. In addition, if ψ has a finite model of size N, then $f(\psi,\alpha)$ has a finite model of size $N + |\alpha|$.

∎

For if Proposition 5.10 holds, then $S_e(P_\beta)$ has the small model property for satisfiability. More specifically, given $\phi \in S_e(P_\beta)$, we can determine a path α and $\psi \in S(P_\beta)$ in linear time, such that $\phi = f(\psi,\alpha)$. In addition, $|\phi| \leq |\psi| + |\alpha|$. If ϕ is satisfiable, then by Proposition 5.10, so is ψ. By Proposition 5.8, ψ has a model of size at most $2^{|\psi|}$. Thus

again by Proposition 5.10, ϕ has a model of size at most $2^{|\psi|} + |\alpha|$, which is no larger than $2^{|\phi|}$.

We next show Proposition 5.10.

Proof: We only prove (2) of the proposition. The proof of (1) is similar.

First notice that if $|\alpha| = 0$, then $f(\psi, \alpha) = \psi$. Obviously, the proposition holds in this case. Hence in the sequel, we assume that $|\alpha| \geq 1$.

Let $\psi = \bigwedge \Sigma \wedge \neg\varphi$, and let

$$R_\alpha = \{\rho \mid \rho \text{ is a path, } \rho \prec_p \alpha\}.$$

Here $\rho \prec_p \alpha$ means that ρ is a proper prefix of α, as described in Chapter 2. The proof of the proposition is carried out as follows.

(1) Suppose that ψ has a finite model $G = (|G|, r^G, E^G)$. We show that $f(\psi, \alpha)$ has a finite model $H = (|H|, r^H, E^H)$, and moreover, the size of H, $size(H)$, is $size(G) + |\alpha|$.

We construct H as follows. For each $\rho \in R_\alpha$, let c_ρ be a distinguished node not in $|G|$. Let

- $|H| = |G| \cup \{c_\rho \mid \rho \in R_\alpha\}$;
- $r^H = c_\epsilon$;
- For all $a, b \in |H|$ and each $K \in E$, $H \models K(a, b)$ iff one of the following conditions is satisfied:
  - there exists $\rho \in R_\alpha$, such that $a = c_\rho$ and $b = c_{\rho \cdot K}$ and $\rho \cdot K \in R_\alpha$; or
  - there exists $\rho \in R_\alpha$, such that $\alpha = \rho \cdot K$ and $a = c_\rho$ and $b = r^G$; or
  - $a, b \in |G|$ and $G \models K(a, b)$.

Obviously, $size(H) = size(G) + |\alpha|$.

To show that $H \models f(\psi, \alpha)$, first observe the following simple facts, which are immediate from the construction of H.

Fact 1: $\{c \mid c \in |H|,\ H \models \alpha(c_\epsilon, c)\} = \{r^G\}$.

Fact 2: For each $a \in |G|$ and each $c \in |H| \setminus |G|$, there exists no path ρ such that $G \models \rho(a, c)$.

Next, we show that $H \models \bigwedge_{\phi \in \Sigma} \delta(\phi, \alpha) \wedge \neg\delta(\varphi, \alpha)$.

First, suppose, for *reductio*, that there exists $\phi \in \Sigma$ such that $H \models \neg\delta(\phi, \alpha)$. Without loss of generality, assume that ϕ is a forward constraint (the argument for the backward case is analogous). Then there exist $a, b, c \in |H|$, such that

$$H \models \alpha(c_\epsilon, a) \wedge pf(\phi)(a, b) \wedge lt(\phi)(b, c) \wedge \neg rt(\phi)(b, c).$$

By Fact 1, $a = r^G$. By Fact 2, $b, c \in |G|$, and moreover, by the construction of H,

$$G \models pf(\phi)(a, b) \wedge lt(\phi)(b, c) \wedge \neg rt(\phi)(b, c).$$

That is, $G \models \neg\phi$. This contradicts the assumption that $G \models \psi$.

Second, since $G \models \psi$, $G \models \neg\varphi$. Without loss of generality, assume that φ is a forward constraint (the argument for the backward case is analogous). Hence there exist $b, c \in |G|$, such that

$$G \models pf(\varphi)(r^G, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

By Fact 1, $H \models \alpha(c_\epsilon, r^G)$. Hence by the construction of H,

$$H \models \alpha(c_\epsilon, r^G) \wedge pf(\varphi)(r^G, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

That is, $H \models \neg\delta(\varphi, \alpha)$.

Hence $H \models f(\psi, \alpha)$. Therefore, H is a finite model of $f(\psi, \alpha)$.

(2) Suppose that $f(\psi, \alpha)$ has a finite model $G = (|G|, r^G, E^G)$. We construct a finite model of ψ.

Without loss of generality, assume that φ is a forward constraint (the proof for the backward case is analogous). Since $G \models \neg\delta(\varphi, \alpha)$, i.e.,

$$G \models \exists\, x\, y\, (\alpha \cdot pf(\varphi)(r^G, x) \wedge lt(\varphi)(x, y) \wedge \neg rt(\varphi)(x, y)),$$

there exist $a, b, c \in |G|$, such that

$$G \models \alpha(r^G, a) \wedge pf(\varphi)(a, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

Let $m = max\{|pf(\phi)| + |lt(\phi)| + |rt(\phi)| \mid \phi \in \Sigma \cup \{\varphi\}\} + 1$, and let $G(a)$ be the m-neighborhood of a in G, as described in Definition 5.5. Clearly, $G(a)$ is a finite σ-structure. We next show that $G(a) \models \psi$.

We first show that $G(a) \models \neg\varphi$. Since $|pf(\varphi)| + |lt(\varphi)| < m$ and $|pf(\varphi)| + |rt(\varphi)| < m$, we have that $b \in |G(a)|$ and $c \in |G(a)|$. Thus by the definition of $G(a)$, we have

$$G(a) \models pf(\varphi)(a, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

That is, $G(a) \models \neg\varphi$.

Second, we show by *reductio* that for each $\phi \in \Sigma$, $G(a) \models \phi$. Suppose that there exists $\phi \in \Sigma$, such that $G(a) \models \neg\phi$. Without loss of generality, assume that ϕ is a forward constraint (the proof for the backward case is analogous). Then there exist $d, e \in |G(a)|$, such that

$$G(a) \models pf(\phi)(a, d) \wedge lt(\phi)(d, e) \wedge \neg rt(\phi)(d, e).$$

Thus by the definition of $G(a)$, we have

$$G \models \alpha(r^G, a) \wedge pf(\phi)(a, d) \wedge lt(\phi)(d, e) \wedge \neg rt(\phi)(d, e).$$

That is, $G \models \neg\delta(\phi, \alpha)$. This contradicts the assumption that $G \models f(\psi, \alpha)$.

This completes the proof of the proposition. ∎

Note that the proof of Proposition 5.10 does not use any special property of P_β, and therefore, still holds for arbitrary recursive subsets of P_c. More specifically, given any recursive subset X of P_c, we can define the function δ for sentences of X in the same way as in Definition 5.3. Similarly, the prefix extended subsets of X can also be defined. Let

$$\begin{aligned} S(X) &= \{\bigwedge \Sigma \wedge \neg\varphi \mid \Sigma \cup \{\varphi\} \text{ is a finite subset of } X\}, \\ S_e(X) &= \{\bigwedge \Sigma \wedge \neg\varphi \mid \Sigma \cup \{\varphi\} \text{ is a prefix extended subset of } X\}. \end{aligned}$$

Define the *prefix extension function from* $S(X)$ *to* $S_e(X)$ as the mapping

$$f_x : S(X) \times Paths \to S_e(X),$$

such that

$$f_x(\bigwedge \Sigma \wedge \neg\varphi,\, \alpha) \mapsto \bigwedge_{\phi\in\Sigma} \delta(\phi, \alpha) \wedge \neg\delta(\varphi, \alpha).$$

It is easy to see that the argument for Proposition 5.10 also serves as a proof of the theorem below.

Theorem 5.11: Let X be a recursive subset of P_c, ψ a sentence in $S(X)$, α a path, and f_x the prefix extension function from $S(X)$ to $S_e(X)$. Then in the context of $\mathcal{SM}$,

1. ψ is satisfiable iff $f_x(\psi, \alpha)$ is satisfiable;
2. ψ is finitely satisfiable iff $f_x(\psi, \alpha)$ is finitely satisfiable. In addition, if ψ has a finite model of size N, then $f_x(\psi, \alpha)$ has a finite model of size $N + |\alpha|$.

■

5.5 Conjunctive path constraints

In this section, we present a mild generalization of P_c, $P_c^\wedge$, and show that the results established in the previous sections also hold for $P_c^\wedge$.

We first define $P_c^\wedge$ as follows.

Definition 5.6: A *conjunctive path constraint* ϕ is an expression of either the *forward* form

$$\forall x\,(\bigwedge_{\alpha\in A} \alpha(r, x) \to \forall y\,(\bigwedge_{\beta\in B} \beta(x, y) \to \gamma(x, y))),$$

or the *backward* form

$$\forall x\,(\bigwedge_{\alpha\in A} \alpha(r, x) \to \forall y\,(\bigwedge_{\beta\in B} \beta(x, y) \to \gamma(y, x))),$$

where A, B are non-empty finite sets of paths, and are denoted by $pf(\phi)$ and $lt(\phi)$, respectively. Here γ is a path, denoted by $rt(\phi)$.

The set of all conjunctive path constraints is denoted by $P_c^\wedge$. ∎

For example, all the constraints given in Chapter 1 are constraints of $P_c^\wedge$. In particular, the constraint (†):

$$\forall x\,(Dept(r,x) \rightarrow \forall y\,((Student(x,y) \wedge Employee(x,y)) \rightarrow TA(x,y)))$$

is also in $P_c^\wedge$, where $A = \{Dept\}$ and $B = \{Student,\ Employee\}$. In Chapter 1, it was demonstrated that constraints of this form are very useful for, among others, describing structural information.

Every path constraint of P_c is a conjunctive path constraint of $P_c^\wedge$. As an immediate corollary of the undecidability results established for P_c, we have the following.

Corollary 5.12: In the context of $\mathcal{SM}$, the implication problem for $P_c^\wedge$ is r.e. complete, and the finite implication problem for $P_c^\wedge$ is co-r.e. complete. ∎

Let $P_f^\wedge$ be the set of all the constraints of $P_c^\wedge$ having the forward form, and let

$$P_+^\wedge = \{\varphi \mid \varphi \in P_c^\wedge,\ rt(\varphi) \neq \epsilon,\ \text{none of the paths in } lt(\varphi) \text{ is } \epsilon\}.$$

Then $P_f \subset P_f^\wedge$ and $P_+ \subset P_+^\wedge$. Therefore, from Theorems 4.2 and 4.3 follow immediately the following corollaries.

Corollary 5.13: In the context of $\mathcal{SM}$, the implication problem for $P_+^\wedge$ is r.e. complete, and the finite implication problem for $P_+^\wedge$ is co-r.e. complete. ∎

Corollary 5.14: In the context of $\mathcal{SM}$, the implication problem for $P_f^\wedge$ is r.e. complete, and the finite implication problem for $P_f^\wedge$ is co-r.e. complete. ∎

We next define fragments of $P_c^\wedge$ analogous to the fragments of P_c described in Section 5.1.

Definition 5.7: A finite subset Σ of $P_c^\wedge$ is called a *prefix restricted subset* of $P_c^\wedge$ iff for all ϕ, ψ in Σ, all the paths in $pf(\phi) \cup pf(\psi)$ have the same length.

The *prefix restricted (finite) implication problem for* $P_c^\wedge$ is the problem of determining,

given any finite prefix restricted subset $\Sigma \cup \{\phi\}$ of $P_c^\wedge$, whether all the (finite) models of Σ are also models of ϕ. ∎

Definition 5.8: A *simple conjunctive path constraint* ϕ is a constraint of $P_c^\wedge$ with $pf(\phi)$ being $\{\epsilon\}$.

A *β-restricted conjunctive path constraint* ϕ is a $P_c^\wedge$ constraint such that for each $\beta \in lt(\phi)$, $|\beta| \leq 1$. That is, either $\beta = \epsilon$, or $\beta = K$ for some $K \in E$.

The set of all simple conjunctive path constraints and all β-restricted conjunctive path constraints is denoted by $P_\beta^\wedge$.

The *(finite) implication problem for* $P_\beta^\wedge$ is the problem of determining, given any finite subset $\Sigma \cup \{\phi\}$ of $P_\beta^\wedge$, whether all the (finite) models of Σ are also models of ϕ. ∎

Definition 5.9: Let ρ be a path and ϕ be a constraint in $P_\beta^\wedge$. The *extension of ϕ with prefix ρ*, denoted by $\delta(\phi, \rho)$, is the constraint in $P_c^\wedge$ defined either by

$$\forall x \, (\bigwedge_{\alpha \in pf(\phi)} \rho \cdot \alpha(r, x) \rightarrow \forall y \, (\bigwedge_{\beta \in lt(\phi)} \beta(x, y) \rightarrow rt(\phi)(x, y)))$$

when ϕ is of the forward form, or by

$$\forall x \, (\bigwedge_{\alpha \in pf(\phi)} \rho \cdot \alpha(r, x) \rightarrow \forall y \, (\bigwedge_{\beta \in lt(\phi)} \beta(x, y) \rightarrow rt(\phi)(y, x)))$$

when ϕ is of the backward form.

Let ρ be a path and Σ be a finite subset of $P_\beta^\wedge$. The *extension of Σ with prefix ρ* is the subset of $P_c^\wedge$ defined by

$$\{\delta(\phi, \rho) \mid \phi \in \Sigma\}.$$

Such a set is called a *prefix extended subset of* $P_\beta^\wedge$.

The *extended (finite) implication problem for* $P_\beta^\wedge$ is the problem of determining, given any prefix extended subset $\Sigma \cup \{\phi\}$ of $P_\beta^\wedge$, whether all the (finite) models of Σ are also models of ϕ. ∎

The following decidability results can be verified analogously to Theorems 5.1, 5.7 and 5.9, respectively.

Corollary 5.15: In the context of $\mathcal{SM}$, the prefix restricted implication and finite implication problems for $P_c^{\wedge}$ are decidable. ■

Corollary 5.16: In the context of $\mathcal{SM}$, the implication and finite implication problems for $P_\beta^{\wedge}$ are decidable. ■

Corollary 5.17: In the context of $\mathcal{SM}$, the extended implication and finite implication problems for $P_\beta^{\wedge}$ are decidable. ■

Chapter 6

Path Constraint Implication in $\mathcal{DM}$

This chapter studies path constraint implication in the context of the deterministic data model $\mathcal{DM}$. More specifically, four path constraint languages and their associated implication and finite implication problems are investigated in $\mathcal{DM}$: P_w (Definition 2.3), P_c (Definition 2.2), and two extensions of P_c, namely, P_c^- (Definition 3.2) and P_c^* (Definition 3.1).

The constraint languages P_w and P_c have been studied for the semistructured data model $\mathcal{SM}$. In [9], it was shown that implication and finite implication of word constraints are finitely axiomatizable and are decidable in PTIME in the context of $\mathcal{SM}$. In Chapter 4, it has been shown that in $\mathcal{SM}$, the implication problem for P_c is r.e. complete and the finite implication problem for P_c is co-r.e. complete (Theorem 4.1). As immediate corollaries of Theorem 4.1, we have the following:

Corollary 6.1: In the context of $\mathcal{SM}$, the implication and finite implication problems for P_c^- are undecidable. ∎

Corollary 6.2: In the context of $\mathcal{SM}$, the implication and finite implication problems for P_c^* are undecidable. ∎

This chapter shows that in the context of $\mathcal{DM}$, Theorem 4.1 and Corollary 6.1 break down. This demonstrates that the determinism condition of $\mathcal{DM}$ simplifies reasoning about path constraints. However, the implication and finite implication problems for P_c^* remain undecidable in the context of $\mathcal{DM}$. This undecidability result shows that the determinism condition of $\mathcal{DM}$ does not trivialize the problem of path constraint implication.

In summary, this chapter establishes a number of complexity results for the following constraint languages in the context of $\mathcal{DM}$:

- The class of word constraints P_w (Section 6.1). We show that the axiomatization developed in [9] is no longer complete for P_w. In light of this breakdown, we present a finite axiomatization for P_w in $\mathcal{DM}$, and moreover, provide a cubic-time algorithm for testing word constraint implication.

- The class of path constraints P_c (Section 6.2). In contrast to Theorem 4.1, we show that in $\mathcal{DM}$, implication and finite implication of P_c constraints are not only decidable in cubic-time, but are also finitely axiomatizable.

- A generalization of P_c, P_c^-, defined in terms of $*$-free regular expressions (Section 6.3). We show that in contrast to Corollary 6.1, the implication and finite implication problems for P_c^- become decidable in the context of $\mathcal{DM}$.

- A generalization of P_c, P_c^*, defined in terms of regular expressions (Section 6.4). We show that the implication and finite implication problems for P_c^* remain undecidable in the context of $\mathcal{DM}$.

Before we prove these complexity results, we first describe an important property of deterministic structures.

Lemma 6.3: Let G be a deterministic structure. Then for any path ρ and node $a \in |G|$, there is at most one node b such that $G \models \rho(a, b)$. ∎

This lemma can be verified by a straightforward induction on $|\rho|$.

6.1 The implication problems for P_w

In this section, we show that in the context of the deterministic data model $\mathcal{DM}$, P_w has the following properties:

- The implication and finite implication problems for P_w are decidable in linear-space.

- P_w is finitely axiomatizable.

- There is an algorithm for testing implication and finite implication of constraints of P_w in cubic-time.

6.1.1 The decidability

We begin with a small model argument for the decidability of the implication and finite implication problems for P_w.

Proposition 6.4: In the context of $\mathcal{DM}$, the implication and finite implication problems for P_w coincide and are decidable in linear-space. ∎

Proof: It suffices to show:

Claim: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_w, and $\phi = \bigwedge \Sigma \wedge \neg\varphi$. If there is a deterministic structure G such that $G \models \phi$, then there exists a deterministic structure H such that $H \models \phi$ and the size of H is at most the length of ϕ.

For if the claim holds, then the satisfiability problem corresponding to the implication problem for P_w has the small model property. Therefore, the implication and finite implication problems for P_w coincide and are decidable in linear-space.

To show the claim, assume that there is a deterministic structure G satisfying ϕ. Recall that a constraint ψ of P_w can be described as

$$\forall\, x\, (lt(\psi)(r,\, x) \rightarrow rt(\psi)(r,\, x)),$$

where $lt(\psi)$ and $rt(\psi)$ are paths, as described in Definition 2.3. Let

$$\begin{aligned} Pts(\phi) &= \{lt(\psi),\, rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\}\}, \\ CloPts(\phi) &= \{\rho \mid \varrho \in Pts(\phi),\, \rho \preceq_p \varrho\}. \end{aligned}$$

Here $\rho \preceq_p \varrho$ stands for that ρ is a prefix of ϱ. Let E_ϕ be the set of edge labels appearing in some path in $Pts(\phi)$. Then we define H to be $(|H|,\, r^H,\, E^H)$ such that

- $|H| = \{a \mid a \in |G|,\, \rho \in CloPts(\phi),\, G \models \rho(r^G,\, a)\}$,
- $r^H = r^G$,

- for all $a, b \in |H|$ and $K \in E$, $H \models K(a,\, b)$ iff $K \in E_\phi$ and $G \models K(a,\, b)$.

It is easy to verify that $H \models \phi$ and H is deterministic, since G has these properties. In addition, by Lemma 6.3, the size of $|H|$ is at most the cardinality of $CloPts(\phi)$, which is at most the length of ϕ. ∎

6.1.2 A finite axiomatization

In the context of $\mathcal{SM}$, it has been shown in [9] that the following inference rules are sound and complete for implication and finite implication of constraints of P_w:

- Reflexivity:

$$\frac{}{\forall x\ (\alpha(r, x) \rightarrow \alpha(r, x))}$$

- Transitivity:

$$\frac{\forall x\ (\alpha(r, x) \rightarrow \beta(r, x)) \quad \forall x\ (\beta(r, x) \rightarrow \gamma(r, x))}{\forall x\ (\alpha(r, x) \rightarrow \gamma(r, x))}$$

- Right-congruence:

$$\frac{\forall x\ (\alpha(r, x) \rightarrow \beta(r, x))}{\forall x\ (\alpha \cdot \gamma(r, x) \rightarrow \beta \cdot \gamma(r, x))}$$

These rules, however, are not complete for P_w in the context of $\mathcal{DM}$. To illustrate this, let α be a path and consider the following constraints of P_w:

$$\begin{aligned} \varphi &= \forall x\, (\epsilon(r,\, x) \rightarrow \alpha(r,\, x)) \\ \phi &= \forall x\, (\alpha(r,\, x) \rightarrow \epsilon(r,\, x)) \end{aligned}$$

By Lemma 6.3, it is easy to verify that $\varphi \models \phi$. However, this implication cannot be derived by using the rules given above.

Next, we show that P_w is still finitely axiomatizable in the context of $\mathcal{DM}$. To do this, we first give the following lemma.

Lemma 6.5: For every finite subset $\Sigma \cup \{\varphi\}$ of P_w,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models_f \varphi.$$

■

Proof: Obviously, if $\Sigma \models \varphi$ then $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi$.

Conversely, if $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi$, then

$$\Sigma \models \exists\, x\, (lt(\varphi)(r,\, x)) \rightarrow \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow rt(\varphi)(r,\, x)).$$

That is,

$$\Sigma \models \forall\, x\, \neg lt(\varphi)(r,\, x) \vee \forall\, x\, (\neg lt(\varphi)(r,\, x) \vee rt(\varphi)(r,\, x)).$$

Since $\forall\, x\, \neg lt(\varphi)(r,\, x) \models \forall\, x\, (\neg lt(\varphi)(r,\, x) \vee rt(\varphi)(r,\, x))$, we have

$$\Sigma \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow rt(\varphi)(r,\, x)).$$

That is, $\Sigma \models \varphi$.

The same proof can be used to show the finite case. ■

Based on this observation, we extend P_w by including constraints of the *existential form:*

$$\exists\, x\, \rho(r,\, x)$$

Here ρ is a path. Constraints of this form enable us to assert the existence of paths. As pointed out by [63, 64], this ability is important for specifying Web link characteristics.

Let

$$P_w^e = P_w \cup \{\exists\, x\, \rho(r,\, x) \mid \rho \text{ is a path}\}.$$

For P_w^e, we consider a set of inference rules, $\mathcal{I}_w$, which consists of the following and Reflexivity, Transitivity, and Right-congruence given above.

- Empty-path:

$$\frac{}{\exists x\, \epsilon(r, x)}$$

- Prefix:

$$\frac{\exists x\ (\alpha \cdot \beta(r, x))}{\exists x\ \alpha(r, x)}$$

- Entail:

$$\frac{\exists x\ \alpha(r, x) \quad \forall x\ (\alpha(r, x) \rightarrow \beta(r, x))}{\exists x\ \beta(r, x)}$$

- Symmetry:

$$\frac{\exists x\ \alpha(r, x) \quad \forall x\ (\alpha(r, x) \rightarrow \beta(r, x))}{\forall x\ (\beta(r, x) \rightarrow \alpha(r, x))}$$

Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_w^e. We use $\Sigma \vdash_{\mathcal{I}_w} \varphi$ to denote that φ is provable from Σ using $\mathcal{I}_w$. That is, there is an $\mathcal{I}_w$-proof of φ from Σ. For example, it is easy to verify the following:

$$\{\exists\, x\ \alpha(r,\ x),\ \forall\, x\,(\alpha(r,\ x) \rightarrow \alpha \cdot \beta(r,\ x)),\ \forall\, x\,(\alpha(r,\ x) \rightarrow \gamma(r,\ x))\}$$
$$\vdash_{\mathcal{I}_w} \forall\, x\,(\alpha \cdot \beta(r,\ x) \rightarrow \gamma(r,\ x))$$
$$\{\exists\, x\ (\alpha \cdot \rho(r,\ x)),\ \forall\, x\,(\alpha(r,\ x) \rightarrow \beta(r,\ x)),\ \forall\, x\,(\beta(r,\ x) \rightarrow \alpha \cdot \rho(r,\ x))\}$$
$$\vdash_{\mathcal{I}_w} \forall\, x\,(\alpha \cdot \rho(r,\ x) \rightarrow \beta(r,\ x))$$

The theorem below shows that $\mathcal{I}_w$ is a finite axiomatization of P_w in the context of $\mathcal{DM}$.

Theorem 6.6: In the context of $\mathcal{DM}$, for every finite subset $\Sigma \cup \{\varphi\}$ of P_w,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\ x))\} \vdash_{\mathcal{I}_w} \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\ x))\} \vdash_{\mathcal{I}_w} \varphi.$$

■

Proof: By Lemma 6.5, we only need to show that

$$\Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\ x))\} \models \varphi \ \text{ iff } \ \Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\ x))\} \vdash_{\mathcal{I}_w} \varphi.$$

Soundness of $\mathcal{I}_w$ can be verified by induction on the lengths of $\mathcal{I}_w$-proofs. For the proof of completeness, it suffices to show

Claim: Let $\Sigma \cup \{\varphi\}$ be any finite subset of P_w and k be any natural number such that

$$k \geq max\{|lt(\psi)|, |rt(\psi)| \mid \psi \in \Sigma \cup \{\varphi\}\}.$$

Then there is a finite deterministic structure G such that

- $G \models \Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\}$, and
- for every path ρ such that $|\rho| \leq k$, if $G \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$, then

$$\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x)).$$

For if the claim holds and suppose that $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi$, then we have $G \models \varphi$ because $G \models \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\}$. In addition, because G is finite, if it is the case where $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models_f \varphi$, then we also have $G \models \varphi$. Thus again by the claim,

$$\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow rt(\varphi)(r,\, x)).$$

Next, we show the claim. To define the structure G described in the claim, let

$$N = \{\rho \mid \rho \text{ is a path},\ |\rho| \leq k,\ \Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r, x)\}.$$

Then the following should be noted:

- By Empty-path in $\mathcal{I}_w$, $\epsilon \in N$.
- By Prefix in $\mathcal{I}_w$, if $\rho \in N$, then all the prefixes of ρ are also in N. That is, N is prefix-closed.

We also define an equivalence relation $\sim$ on N as follows:

$$\rho \sim \varrho \ \text{ iff } \ \Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x)).$$

It should be noted that for all $\rho, \varrho \in N$, $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x))$ iff $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (\varrho(r,\, x) \rightarrow \rho(r,\, x))$, by Symmetry in $\mathcal{I}_w$.

Let $[\rho]$ be the equivalence class of ρ with respect to $\sim$. For each $\rho \in N$, let $o([\rho])$ be a distinct node. We then define G to be $(|G|,\, r^G,\, E^G)$, where

- $|G| = \{o([\rho]) \mid \rho \in N\}$,
- $r^G = o([\epsilon])$,
- for every $K \in E$ and $o([\rho]), o([\varrho]) \in |G|$, $G \models K(o([\rho]),\, o([\varrho]))$ iff $[\varrho] = [\rho \cdot K]$. This is well-defined by Transitivity, Right-congruence and Symmetry in $\mathcal{I}_w$.

We next show that G is indeed the structure described in the Claim.

(1) G is a finite deterministic structure.

It should be noted that N is finite. Therefore, $|G|$ is finite. In addition, G is deterministic because of Symmetry, Transitivity and Right-congruence in $\mathcal{I}_w$.

(2) $G \models \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\}$.

It suffices to show

Claim 1: For every $\rho \in N$ and path ϱ such that $|\varrho| \leq k$, $G \models \varrho(r^G,\, o([\rho]))$ iff $\varrho \in [\rho]$.

For if Claim 1 holds, then by $lt(\varphi) \in N$, we have that $G \models lt(\varphi)(r^G,\, o([lt(\varphi)])$. That is,

$$G \models \exists\, x\, (lt(\varphi)(r,\, x)).$$

In addition, for every $\phi \in \Sigma$, if there exists $a \in |G|$ such that $G \models lt(\phi)(r^G,\, a)$, then by Claim 1 and the fact that G is deterministic, $lt(\phi) \in N$ and $a = o([lt(\phi)])$. By Entail in $\mathcal{I}_w$, we have $rt(\phi) \in N$, and moreover, $rt(\phi) \sim lt(\phi)$. Therefore, again by Claim 1, we have $G \models rt(\phi)(r^G,\, o([lt(\phi)]))$. Thus $G \models \phi$. Hence $G \models \Sigma$.

We show Claim 1 by induction on $|\varrho|$.

Base case: $\varrho = \epsilon$.

Clearly, $G \models \epsilon(o([\epsilon]),\, o([\rho]))$ iff $[\rho] = [\epsilon]$ iff $\epsilon \in [\rho]$.

Inductive step: Assume Claim 1 for ϱ. We next show that Claim 1 also holds for $\varrho \cdot K$.

If $G \models \varrho \cdot K(r^G,\, o([\rho]))$, then by the induction hypothesis and Lemma 6.3, we have

$$G \models \varrho(r^G,\, o([\varrho])) \wedge K(o([\varrho]),\, o([\rho])).$$

By the definition of E^G, $G \models K(o([\varrho]),\, o([\rho]))$ iff $[\rho] = [\varrho \cdot K]$. Therefore, $\varrho \cdot K \in [\rho]$.

Conversely, if $\varrho \cdot K \in [\rho]$, then by Prefix in $\mathcal{I}_w$, we have $\varrho \in N$. By the induction hypothesis,

$$G \models \varrho(r^G, o([\varrho])).$$

By the definition of E^G, $G \models K(o([\varrho]), o([\varrho \cdot K]))$. Moreover, by $\varrho \cdot K \in [\rho]$, we have $[\varrho \cdot K] = [\rho]$ and therefore, $G \models \varrho \cdot K(r^G, o([\rho]))$.

(3) For any ρ such that $|\rho| \leq k$, $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$ if $G \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$.

Note that $lt(\varphi) \in N$. By Claim 1,

$$G \models lt(\varphi)(r,\, o([lt(\varphi)])).$$

Thus if $G \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$, then $G \models \rho(r^G,\, o([lt(\varphi)]))$. Again by Claim 1, we have $\rho \in [lt(\varphi)]$. That is, $\rho \sim lt(\varphi)$. Hence by the definition of $\sim$ and Symmetry in $\mathcal{I}_w$, we have

$$\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x)).$$

This completes the proof of Theorem 6.6. ∎

In fact, $\mathcal{I}_w$ is a finite axiomatization of P^e_w.

Theorem 6.7: In the context of $\mathcal{DM}$, for every finite subset $\Sigma \cup \{\varphi\}$ of P^e_w, if $\varphi \in P_w$, then

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi.$$

Otherwise, i.e., when φ is a constraint of the existential form,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_w} \varphi$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_w} \varphi.$$

∎

Proof: Soundness of $\mathcal{I}_w$ can be verified by induction on the lengths of $\mathcal{I}_w$-proofs. For the proof of completeness, we consider two cases.

(1) $\varphi \in P_w$.

The proof for this case is similar to the argument for Theorem 6.6.

(2) $\varphi \notin P_w$. That is, $\varphi = \exists\, x\ \alpha(r, x)$.

In this case, it suffices to show

Claim: Let $\Sigma \cup \{\varphi\}$ be any finite subset of P_w^e and k be any natural number such that

$$k \geq max\{|lt(\psi)|, |rt(\psi)| \mid \psi \in \Sigma \cap P_w\} \quad \text{and} \quad k \geq max\{|\rho| \mid \exists\, x\ \rho(r,\, x) \in \Sigma \cup \{\varphi\}\}.$$

Then there is a finite deterministic structure G such that

- $G \models \Sigma$, and
- for every path ρ such that $|\rho| \leq k$, if $G \models \exists\, x\ \rho(r,\, x)$, then $\Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r,\, x)$.

For if the claim holds and $\Sigma \models \exists\, x\ \alpha(r,\, x)$, then we have $G \models \exists\, x\ \alpha(r,\, x)$, since $G \models \Sigma$. In addition, since G is finite, if $\Sigma \models_f \exists\, x\ \alpha(r,\, x)$, then we also have $G \models \exists\, x\ \alpha(r,\, x)$. Thus again by Claim, $\Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \alpha(r,\, x)$. That is, $\Sigma \vdash_{\mathcal{I}_w} \varphi$.

Next, we show the claim. We first define the following:

$$\begin{aligned} N &= \{\rho \mid \rho \text{ is a path, } |\rho| \leq k,\ \Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r, x)\} \\ \rho \sim \varrho &\quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x)) \end{aligned}$$

In addition, we define $[\rho]$, $o([\rho])$ and G as in the proof of Theorem 6.6. Then using the same argument given in the proof of Theorem 6.6, we can verify

Claim 1: For every $\rho \in N$ and path ϱ such that $|\varrho| \leq k$, $G \models \varrho(r^G,\, o([\rho]))$ iff $\varrho \in [\rho]$.

Using Claim 1, we can show the following.

Statement 1: $G \models \Sigma$.

Let ϕ be any constraint in Σ. If ϕ is $\exists\, x\ \rho(r,\, x)$, then we have $\rho \in N$, and in addition, by Claim 1, $G \models \rho(r^G,\, o([\rho]))$. That is, $G \models \phi$.

If $\phi \in P_w$, then by using the same argument given in the proof of Theorem 6.6, it can also be shown that $G \models \phi$.

Statement 2: For every path ρ such that $|\rho| \leq k$, if $G \models \exists\, x\ \rho(r,\, x)$, then $\Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r,\, x)$.

If $G \models \exists\, x\ \rho(r,\, x)$, then there exists $\varrho \in N$ such that $G \models \rho(r^G,\, o([\varrho]))$. By Claim 1, $\rho \in [\varrho]$. By the definition of N and $\sim$, we have $\rho \in N$ and in addition,

$$\Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r,\, x).$$

This completes the proof of Theorem 6.7. ∎

6.1.3 An algorithm

Next, we present an algorithm for testing word constraint implication. This algorithm takes as input a finite subset Σ of P_w and a path α. It returns as output a deterministic structure G having the following properties: there is $o \in |G|$ such that $G \models \alpha(r^G,\, o)$, and moreover, for any path β,

$$G \models \beta(r^G, o) \ \text{ iff } \ \Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\alpha(r,\, x) \rightarrow \beta(r,\, x)).$$

By Theorem 6.6, this algorithm can be used for testing implication and finite implication of word constraints.

The algorithm (Algorithm 6.1) is given in Table 6.1. The procedure $merge(a,\, b)$ used in the algorithm is shown in Table 6.2. The structure G computed by Algorithm 6.1 can be naturally extended to a σ-structure by letting $K^G = \emptyset$ for any $K \in E \setminus E_\phi$.

It should be noted that the rationale behind step 4 (1) of Algorithm 6.1 is Lemma 6.3. In a deterministic graph G, for any path ρ, if there is $o \in |G|$ such that $G \models \rho(r^G,\, o)$, then o is unique. As a result, every constraint in Σ is used at most once by the algorithm. In general, this does not hold in graphs of $\mathcal{SM}$. It is because of this property that Algorithm 6.1 has low complexity.

For the complexity of the algorithm, the following should be noted. Let n_E be the cardinality of E_ϕ, n_G the size of $|G|$, n the length of Σ and α, and n_Σ the cardinality of Σ.

- $n_E \leq n$, $n_G \leq n$ and $n_\Sigma \leq n$.

Algorithm 6.1:

Input: a finite subset Σ of P_w and a path α
Output: the structure G described above

1. E_ϕ := the set of edge labels appearing in either α or some path in constraints of Σ;
2. $Rules := \Sigma$;
3. $G := (|G|, r^G, E_\phi^G)$, where
   - $|G| = \{o(\rho) \mid \rho \preceq_p \alpha,\ o(\rho)$ is a distinct node$\}$,
   - $r^G = o(\epsilon)$,
   - E_ϕ^G is populated such that $G \models K(o(\rho), o(\varrho))$ iff $\varrho = \rho \cdot K$;
4. repeat until no further change:
   if $\forall x\,(\rho(r, x) \rightarrow \varrho(r, x)) \in \Sigma$ and there is $o_\rho \in |G|$ such that $G \models \rho(r^G, o_\rho)$ then
   (1) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \rightarrow \varrho(r, x))\}$;
   (2) for each $\gamma \cdot K \preceq_p \varrho$ do
   if there is no $o \in |G|$ such that $G \models \gamma \cdot K(r^G, o)$ then
   (i) add to $|G|$ a distinct node $o_{\gamma \cdot K}$;
   (ii) add to E_ϕ^G an edge labeled K from o_γ to $o_{\gamma \cdot K}$,
   where $o_\gamma \in |G|$ such that $G \models \gamma(r^G, o_\gamma)$;
   (3) $merge(o_\rho, o_\varrho)$;
5. output G.

Table 6.1: An algorithm for testing word constraint implication in $\mathcal{DM}$

procedure $merge(a, b)$

1. for each $K \in E_\phi$ do
   if there is $o \in |G|$ such that $G \models K(o, b)$ then
   (1) delete from E_ϕ^G the edge labeled K from o to b;
   (2) add to E_ϕ^G an edge labeled K from o to a;
2. for each $K \in E_\phi$ do
   if there is $o_b \in |G|$ such that $G \models K(b, o_b)$ then
   (1) delete from E_ϕ^G the edge labeled K from b to o_b;
   (2) add to E_ϕ^G an edge labeled K from a to o_b;
   (3) if there is $o_a \in |G|$ such that $G \models K(a, o_a)$ and $o_a \neq o_b$ then
   $merge(o_a, o_b)$;
3. $|G| := |G| \setminus \{b\}$;

Table 6.2: Procedure *merge* used in Algorithm 6.1

- Step 4(1), (2) and (3) are executed at most n_Σ times.
- Testing whether $G \models \rho(r^G, o_\rho)$ in step 4 can be done in at most $O(n_G\,|\rho|)$ time. Therefore, it can be done in $O(n^2)$ time. By using appropriate data structure, e.g., (variable length) array indexed by edge labels in E_ϕ, this can be done in $O(|\rho|)$ time, i.e., $O(n)$ time.
- The procedure *merge* is executed at most n_G times. Each step takes $O(n_E\ n_G)$ time. Hence the total cost of executing *merge* is $O(n_G^2\, n_E)$, i.e., $O(n^3)$. Again, by using appropriate data structure, this can be done in $O(n^2)$ time.

Therefore, Algorithm 6.1 runs in $O(n^3)$ time. In addition, when implemented using appropriate data structures, this algorithm runs in $O(n^2)$ time.

Next, we show that Algorithm 6.1 is correct.

Proposition 6.8: Given a finite subset Σ of P_w and a path α, Algorithm 6.1 computes a finite deterministic structure G having the following property: there exists $o \in |G|$, such that $G \models \alpha(r^G,\, o)$, and in addition, for any path β,

$$G \models \beta(r^G, o) \;\text{ iff }\; \Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\alpha(r,\, x) \to \beta(r,\, x)).$$

■

Proof: The step 4 of Algorithm 6.1 ensures that $G \models \Sigma$, taking advantage of the fact that G is deterministic and because of Lemma 6.3. In addition, step 3 ensures that there is $o \in |G|$, such that $G \models \alpha(r^G,\, o)$. Therefore, if $\Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\alpha(r,\, x) \to \beta(r,\, x))$, then by Theorem 6.6, we have

$$G \models \beta(r^G, o).$$

Conversely, by a straightforward induction on the number of steps in the construction of G by the algorithm, we can show that for all paths ρ and ϱ, if there is $a \in |G|$ such that $G \models \rho(r^G,\, a) \wedge \varrho(r^G,\, a)$, then

$$\Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \to \varrho(r,\, x)).$$

Indeed, each step of the construction in fact corresponds to applications of some rules in $\mathcal{I}_w$. For example, step 4(2) corresponds to an application of Prefix, and *merge* corresponds to

applications of Transitivity, Right-Congruence and Symmetry in $\mathcal{I}_w$. Thus if $G \models \beta(r^G, o)$, then we have $\Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\alpha(r,\, x) \rightarrow \beta(r,\, x))$. ∎

From Proposition 6.8, Algorithm 6.1, and Theorem 6.6, the corollary below follows immediately.

Corollary 6.9: In the context of $\mathcal{DM}$, the implication and finite implication problems for P_w are decidable in cubic-time. ∎

6.2 The implication problems for P_c

This section generalizes the results established in the last section to P_c. In contrast to the undecidability of the implication and finite implication problems for P_c in $\mathcal{SM}$, we show that in the context of $\mathcal{DM}$, P_c has the following properties:

- The implication and finite implication problems for P_c coincide and are decidable in linear-space.
- There is a finite axiomatization for (finite) implication of P_c constraints.
- There is a cubic-time algorithm for testing (finite) implication of P_c constraints.

6.2.1 The decidability

With slight modification, the small model argument for Proposition 6.4 is also applicable to the proposition below.

Proposition 6.10: In the context of $\mathcal{DM}$, the implication and finite implication problems for P_c coincide and are decidable in linear-space. ∎

Proof: As in the proof of Proposition 6.4, it suffices to show:

Claim: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c, and $\phi = \bigwedge \Sigma \wedge \neg\varphi$. If there is a deterministic structure G such that $G \models \phi$, then there exists a deterministic structure H such that

$H \models \phi$ and the size of H is at most the length of ϕ.

To show the claim, assume that ϕ has a deterministic model G. Let

$$\begin{aligned} Pts(\phi) &= \{pf(\psi)\cdot lt(\psi),\ pf(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the forward form}\} \\ &\quad \cup \{pf(\psi)\cdot lt(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the backward form}\}, \\ CloPts(\phi) &= \{\rho \mid \varrho \in Pts(\phi),\ \rho \preceq_p \varrho\}. \end{aligned}$$

Let E_ϕ be the set of edge labels appearing in some path in $Pts(\phi)$. Then we define H in the same way as in the proof of Proposition 6.4. It is easy to verify that H is a deterministic structure, $H \models \phi$, and in addition, by Lemma 6.3, the size of $|H|$ is at most the cardinality of $CloPts(\phi)$, which is at most the length of ϕ. ∎

6.2.2 A finite axiomatization

Before we present a finite axiomatization for P_c, we first study basic properties of constraints of P_c in $\mathcal{DM}$.

Lemma 6.11: Let φ be a forward constraint of P_c:

$$\varphi = \forall x\,(\alpha(r,\ x) \rightarrow \forall y\,(\beta(x,\ y) \rightarrow \gamma(x,\ y))),$$

and ψ be a constraint of P_w:

$$\psi = \forall x\,(\alpha\cdot\beta(r,\ x) \rightarrow \alpha\cdot\gamma(r,\ x)).$$

Then for any deterministic structure G, $G \models \varphi$ iff $G \models \psi$. ∎

Proof: If $G \models \neg\psi$, then there is $b \in |G|$ such that

$$G \models \alpha\cdot\beta(r^G,\ b) \wedge \neg\alpha\cdot\gamma(r^G,\ b).$$

Thus there exists $a \in |G|$ such that $G \models \alpha(r^G,\ a) \wedge \beta(a,\ b)$. In addition, $G \models \neg\gamma(a,\ b)$ since otherwise $G \models \alpha\cdot\gamma(r^G,\ b)$. Hence there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\ a) \wedge \beta(a,\ b) \wedge \neg\gamma(a,\ b).$$

Thus $G \models \neg\varphi$.

Conversely, if $G \models \neg\varphi$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(a,\, b).$$

By Lemma 6.3, a is the unique node such that $G \models \alpha(r^G,\, a)$. Thus

$$G \models \alpha \cdot \beta(r^G,\, b) \wedge \neg\alpha \cdot \gamma(r^G,\, b).$$

That is, $G \models \neg\psi$. ∎

Lemma 6.12: Let φ be a backward constraint of P_c:

$$\varphi = \forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x))),$$

and ψ be a constraint of P_w:

$$\psi = \forall\, x\, (\alpha(r,\, x) \rightarrow \alpha \cdot \beta \cdot \gamma(r,\, x)).$$

Then for any deterministic structure G, if $G \models \exists\, x\, (\alpha \cdot \beta(r,\, x))$, then $G \models \varphi$ iff $G \models \psi$. ∎

Proof: Assume that $G \models \exists\, x\, (\alpha \cdot \beta(r,\, x))$. Then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b).$$

By Lemma 6.3, a is the unique node such that $G \models \alpha(r^G,\, a)$, and b is the unique node such that $G \models \beta(a,\, b)$.

If $G \models \neg\psi$, then

$$G \models \alpha(r^G,\, a) \wedge \neg\alpha \cdot \beta \cdot \gamma(r^G,\, a).$$

Clearly, $G \models \neg\gamma(b,\, a)$ since otherwise $G \models \alpha \cdot \beta \cdot \gamma(r^G,\, a)$. Therefore,

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(b,\, a).$$

That is, $G \models \neg\varphi$.

Conversely, if $G \models \neg\varphi$, then there are $a', b' \in |G|$ such that

$$G \models \alpha(r^G,\, a') \wedge \beta(a',\, b') \wedge \neg\gamma(b',\, a').$$

By Lemma 6.3, $a' = a$ and $b' = b$. In addition, $G \models \neg\alpha \cdot \beta \cdot \gamma(r^G, a)$ since otherwise $G \models \gamma(b, a)$. Hence

$$G \models \alpha(r^G, a) \wedge \neg\alpha \cdot \beta \cdot \gamma(r^G, a).$$

Thus $G \models \neg\psi$. ∎

Lemma 6.13: For every finite subset $\Sigma \cup \{\varphi\}$ of P_c,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models_f \varphi.$$

∎

Proof: Obviously, if $\Sigma \models \varphi$ then $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi$.

Conversely, if $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi$, then

$$\Sigma \models \exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)) \rightarrow \varphi.$$

That is,

$$\Sigma \models \forall\, x\, \neg pf(\varphi) \cdot lt(\varphi)(r,\, x) \vee \varphi.$$

Note that φ is of either the forward form:

$$\forall\, x\, (\neg pf(\varphi)(r,\, x) \vee \forall\, y\, (\neg lt(\varphi)(x,\, y) \vee rt(\varphi)(x,\, y))),$$

or the backward form:

$$\forall\, x\, (\neg pf(\varphi)(r,\, x) \vee \forall\, y\, (\neg lt(\varphi)(x,\, y) \vee rt(\varphi)(y,\, x))).$$

Since $\forall\, x\, (\neg pf(\varphi) \cdot lt(\varphi)(r,\, x)) \models \forall\, x \forall\, y\, (\neg pf(\varphi)(r,\, x) \vee \neg lt(\varphi)(x,\, y))$, we have

$$\forall\, x\, (\neg pf(\varphi) \cdot lt(\varphi)(r,\, x)) \models \varphi.$$

Hence $\Sigma \models \varphi$.

The same proof is also applicable to the case of finite implication. ∎

Based on Lemma 6.13, we extend P_c by including constraints of the existential form as follows:

$$P_c^e = P_c \cup \{\exists\, x\, \rho(r,\, x) \mid \rho \text{ is a path}\}.$$

As mentioned in the last section, constraints of the existential form assert existence of paths.

For P_c^e, we consider a set of inference rules, $\mathcal{I}_c$, which consists of the following and those in $\mathcal{I}_w$ given in the last section. Note that the inference rules below are sound in $\mathcal{DM}$ because of Lemmas 6.11 and 6.12.

- Forward-to-word:
$$\frac{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(x,y)))}{\forall x\ (\alpha \cdot \beta(r,x) \rightarrow \alpha \cdot \gamma(r,x))}$$
- Word-to-forward:
$$\frac{\forall x\ (\alpha \cdot \beta(r,x) \rightarrow \alpha \cdot \gamma(r,x))}{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(x,y)))}$$
- Backward-to-word:
$$\frac{\exists x\ (\alpha \cdot \beta(r,x)) \quad \forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(y,x)))}{\forall x\ (\alpha(r,x) \rightarrow \alpha \cdot \beta \cdot \gamma(r,x))}$$
- Word-to-backward:
$$\frac{\exists x\ (\alpha \cdot \beta(r,x)) \quad \forall x\ (\alpha(r,x) \rightarrow \alpha \cdot \beta \cdot \gamma(r,x))}{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(y,x)))}$$

Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c^e. We use $\Sigma \vdash_{\mathcal{I}_c} \varphi$ to denote that φ is provable from Σ using $\mathcal{I}_c$. That is, there is an $\mathcal{I}_c$-proof of φ from Σ.

Similar to Theorem 6.6, the following theorem shows that $\mathcal{I}_c$ is indeed a finite axiomatization of P_c.

Theorem 6.14: In the context of $\mathcal{DM}$, for every finite subset $\Sigma \cup \{\varphi\}$ of P_c,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\ x))\} \vdash_{\mathcal{I}_c} \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\ x))\} \vdash_{\mathcal{I}_c} \varphi.$$

■

Proof: By Lemma 6.13, we only need to show

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\ x))\} \models \varphi \ \text{ iff } \ \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\ x))\} \vdash_{\mathcal{I}_c} \varphi.$$

The proof is similar to the proof of Theorem 6.6. Soundness of $\mathcal{I}_c$ can be verified by induction on the lengths of $\mathcal{I}_c$-proofs. For the proof of completeness, it suffices to show

Claim: Let $\Sigma \cup \{\varphi\}$ be any finite subset of P_c and k be any natural number such that

$$k \geq max\{|pf(\psi)| + |lt(\psi)| + |rt(\psi)| \quad | \quad \psi \in \Sigma \cup \{\varphi\}\}.$$

Then there is a finite deterministic structure G such that

1. $G \models \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r^G,\, x))\}$,
2. for every path ρ such that $|\rho| \leq k - |pf(\varphi) \cdot lt(\varphi)|$,
   - if $G \models \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(x,\, y)))$, then

     $$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \quad \vdash_{\mathcal{I}_c} \quad \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(x,\, y)));$$

   - if $G \models \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x)))$, then

     $$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \quad \vdash_{\mathcal{I}_c} \quad \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x))).$$

To see why this claim suffices, suppose that $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi$. Then by $G \models \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\}$, we have $G \models \varphi$. In addition, since G is finite, if it is the case where $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models_f \varphi$, then we also have $G \models \varphi$. Thus again by the claim, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi.$$

Next, we show the claim. Let

$$N = \{\rho \mid \rho \text{ is a path, } |\rho| \leq k,\ \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_c} \exists\, x\, \rho(r, x)\}.$$

Recall the equivalence relation $\sim$ on N defined in the proof of Theorem 6.6. We also use $[\rho]$ to denote the equivalence class of ρ with respect to $\sim$. For each $\rho \in N$, we create a distinct node $o([\rho])$. Let G be the structure defined in the proof of Theorem 6.6. Using

the same proof, we can show that G is a finite deterministic structure. In addition, we can also show the following claim:

Claim 1: For every $\rho \in N$ and path ϱ such that $|\varrho| \leq k$, $G \models \varrho(r^G, o([\rho]))$ iff $\varrho \in [\rho]$.

Using Claim 1, we show the following.

(1) $G \models \exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r^G, x))$.

Clearly, $pf(\varphi) \cdot lt(\varphi) \in N$. Thus by Claim 1,

$$G \models pf(\varphi) \cdot lt(\varphi)(r^G, o([pf(\varphi) \cdot lt(\varphi)])).$$

(2) $G \models \Sigma$.

Suppose, for *reductio*, that there is $\psi \in \Sigma$ such that $G \models \neg\psi$. Then we show that the assumption leads to a contradiction.

If ψ is a forward constraint $\forall\, x\, (\alpha(r, x) \rightarrow \forall\, y\, (\beta(x, y) \rightarrow \gamma(x, y)))$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(a, b).$$

Thus by Lemma 6.3 and Claim 1, we have $\alpha \cdot \beta \in N$, $a = o([\alpha])$ and $b = o([\alpha \cdot \beta])$. By Forward-to-word and Entail in $\mathcal{I}_c$, we have $\alpha \cdot \gamma \in N$ and moreover,

$$\alpha \cdot \beta \sim \alpha \cdot \gamma.$$

Therefore, again by Claim 1, we have $G \models \alpha \cdot \gamma(r^G, o([\alpha \cdot \beta]))$. By Lemma 6.3, we have

$$G \models \gamma(o([\alpha]), o([\alpha \cdot \beta])).$$

This contradicts the assumption.

If ψ is a backward constraint $\forall\, x\, (\alpha(r, x) \rightarrow \forall\, y\, (\beta(x, y) \rightarrow \gamma(y, x)))$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(b, a).$$

Again by Lemma 6.3 and Claim 1, we have $\alpha \cdot \beta \in N$, $a = o([\alpha])$ and $b = o([\alpha \cdot \beta])$. That is,

$$G \models \exists\, x\, (\alpha \cdot \beta(r, x)).$$

By Backward-to-word and Entail, we have $\alpha \cdot \beta \cdot \gamma \in N$ and moreover,

$$\alpha \sim \alpha \cdot \beta \cdot \gamma.$$

Therefore, again by Claim 1, we have $G \models \alpha \cdot \beta \cdot \gamma(r^G,\, o([\alpha]))$. By Lemma 6.3, we have

$$G \models \gamma(o([\alpha \cdot \beta]),\, o([\alpha])).$$

This again contradicts the assumption.

Thus $G \models \psi$. Hence $G \models \Sigma$.

(3) G has the property described by (2) of Claim.

Let ρ be a path such that $|\rho| \leq k - |pf(\varphi) \cdot lt(\varphi)|$.

If $G \models \forall x\,(pf(\varphi)(r,\, x) \to \forall y\,(lt(\varphi)(x,\, y) \to \rho(x,\, y)))$, then by Lemma 6.11,

$$G \models \forall x\,(pf(\varphi) \cdot lt(\varphi)(r,\, x) \to pf(\varphi) \cdot \rho(r,\, x)).$$

By $pf(\varphi) \cdot lt(\varphi) \in N$ and Claim 1, we have

$$G \models pf(\varphi) \cdot \rho(r^G,\, o([pf(\varphi) \cdot lt(\varphi)])),$$

and moreover,

$$pf(\varphi) \cdot lt(\varphi) \sim pf(\varphi) \cdot \rho.$$

Thus by the definition of $\sim$ and Symmetry in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists x\,(pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall x\,(pf(\varphi) \cdot lt(\varphi)(r,\, x) \to pf(\varphi) \cdot \rho(r^G,\, x)).$$

By Word-to-forward in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists x\,(pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall x\,(pf(\varphi)(r,\, x) \to \forall y\,(lt(\varphi)(x,\, y) \to \rho(x,\, y))).$$

If $G \models \forall x\,(pf(\varphi)(r,\, x) \to \forall y\,(lt(\varphi)(x,\, y) \to \rho(y,\, x)))$, then by $G \models \exists(pf(\varphi) \cdot lt(\varphi)(r,\, x))$ and Lemma 6.12, we have

$$G \models \forall x\,(pf(\varphi)(r,\, x) \to pf(\varphi) \cdot lt(\varphi) \cdot \rho(r,\, x)).$$

Since $pf(\varphi) \cdot lt(\varphi) \in N$, by Prefix in $\mathcal{I}_c$, we have $pf(\varphi) \in N$. By Claim 1, we have

$$G \models pf(\varphi) \cdot lt(\varphi) \cdot \rho(r^G,\, o([pf(\varphi)])),$$

and moreover,

$$pf(\varphi) \sim pf(\varphi) \cdot lt(\varphi) \cdot \rho.$$

Thus by the definition of $\sim$ and Symmetry in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi) \rightarrow pf(\varphi) \cdot lt(\varphi) \cdot \rho(r,\, x)).$$

By Word-to-backward in $\mathcal{I}_c$ and $G \models \exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))$, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x))).$$

This completes the proof of Claim, and therefore, the proof of Theorem 6.14. ∎

In addition, it can be shown that $\mathcal{I}_c$ is also a finite axiomatization of P_c^e.

Theorem 6.15: In the context of $\mathcal{DM}$, for every finite subset $\Sigma \cup \{\varphi\}$ of P_c^e, if $\varphi \in P_c$, then

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi.$$

Otherwise, i.e., when φ is an existential constraints,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_c} \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_c} \varphi.$$

∎

The proof of this theorem is similar to the proof of Theorem 6.14.

6.2.3 An algorithm

Next, we present an algorithm for testing implication of constraints of P_c. This algorithm takes as input a finite subset Σ of P_c and two paths α and β. It computes a deterministic

structure G such that there are $a, b \in |G|$, $G \models \alpha(r^G, a) \wedge \beta(a, b)$, and in addition, for any path γ,

$$G \models \gamma(a, b) \text{ iff } \Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y))),$$
$$G \models \gamma(b, a) \text{ iff } \Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))).$$

By Theorem 6.14, this algorithm can be used for testing implication and finite implication of P_c constraints.

The algorithm (Algorithm 6.2) is shown in Table 6.3, which uses procedure *merge* given in Table 6.2. The structure G computed by the algorithm can be extended to a σ-structure by letting $K^G = \emptyset$ for any $K \in E \setminus E_\phi$.

Similar to the analysis of Algorithm 6.1 given in the last section, it can also be shown that Algorithm runs in $O(n^3)$ time, where n is the length of Σ and $\alpha \cdot \beta$. In addition, when implemented using appropriate data structures, the algorithm runs in $O(n^2)$ time.

The proposition below shows that Algorithm 6.2 is correct.

Proposition 6.16: Given a finite subset Σ of P_c and paths α, β, Algorithm 6.2 computes a finite deterministic structure G having the following property: there are $a, b \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$, and in addition, for any path γ,

$$G \models \gamma(a, b) \text{ iff } \Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y))),$$
$$G \models \gamma(b, a) \text{ iff } \Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))).$$

■

Proof: The step 4 of Algorithm 6.2 ensures that $G \models \Sigma$, taking advantage of the fact that G is deterministic and by using Lemma 6.3. In addition, step 3 ensures that there are $a, b \in |G|$, such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b).$$

Thus if $\Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y)))$, then by Theorem 6.14,

$$G \models \gamma(a, b).$$

Algorithm 6.2:

Input: a finite subset Σ of P_c and paths α, β
Output: the structure G described above

1. E_ϕ := the set of edge labels appearing in either $\alpha \cdot \beta$ or some path in constraints of Σ;

2. $Rules := \Sigma$;

3. $G := (|G|, r^G, E_\phi^G)$, where
   - $|G| = \{o(\rho) \mid \rho \preceq_p \alpha \cdot \beta,\ o(\rho)$ is a distinct node$\}$,
   - $r^G = o(\epsilon)$,
   - E_ϕ^G is populated such that $G \models K(o(\rho), o(\varrho))$ iff $\varrho = \rho \cdot K$;

4. repeat until no further change:
   (1) if $\forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(x, y))) \in \Sigma$ and there are $o_\rho, o_{\rho\cdot\varrho} \in |G|$ such that $G \models \rho(r^G, o_\rho) \wedge \varrho(o_\rho, o_{\rho\cdot\varrho})$ then
      (i) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(x, y)))\}$;
      (ii) for each $\xi \cdot K \preceq_p \zeta$ do
         if there is no $o \in |G|$ such that $G \models \xi \cdot K(o_\rho, o)$ then
            (a) add to $|G|$ a distinct node $o_{\rho\cdot\xi\cdot K}$;
            (b) add to E_ϕ^G an edge labeled K from $o_{\rho\cdot\xi}$ to $o_{\rho\cdot\xi\cdot K}$,
               where $o_{\rho\cdot\xi} \in |G|$ such that $G \models \xi(o_\rho, o_{\rho\cdot\xi})$;
      (iii) $merge(o_{\rho\cdot\varrho}, o_{\rho\cdot\zeta})$;
   (2) if $\forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(y, x))) \in \Sigma$ and there are $o_\rho, o_{\rho\cdot\varrho} \in |G|$ such that $G \models \rho(r^G, o_\rho) \wedge \varrho(o_\rho, o_{\rho\cdot\varrho})$ then
      (i) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(y, x)))\}$;
      (ii) for each $\xi \cdot K \preceq_p \zeta$ do
         if there is no $o \in |G|$ such that $G \models \xi \cdot K(o_{\rho\cdot\varrho}, o)$ then
            (a) add to $|G|$ a distinct node $o_{\rho\cdot\varrho\cdot\xi\cdot K}$;
            (b) add to E_ϕ^G an edge labeled K from $o_{\rho\cdot\varrho\cdot\xi}$ to $o_{\rho\cdot\varrho\cdot\xi\cdot K}$,
               where $o_{\rho\cdot\varrho\cdot\xi} \in |G|$ such that $G \models \xi(o_{\rho\cdot\varrho}, o_{\rho\cdot\varrho\cdot\xi})$;
      (iii) $merge(o_\rho, o_{\rho\cdot\varrho\cdot\zeta})$;

5. output G.

Table 6.3: An algorithm for testing path constraint implication in $\mathcal{DM}$

Similarly, if $\Sigma \cup \{\exists\, x\, (\alpha \cdot \beta(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x)))$, then

$$G \models \gamma(b,\, a).$$

Conversely, by a straightforward induction on the number of steps in the construction of G by the algorithm, we can show that for all paths ρ and ϱ, if there exists a node $o \in |G|$ such that $G \models \rho(r^G,\, o) \wedge \varrho(r^G,\, o)$, then $\Sigma \cup \{\exists\, x\, (\alpha \cdot \beta(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x))$. Indeed, each step of the construction in fact corresponds to applications of some rules in $\mathcal{I}_c$. For example, step 4 (1) corresponds to an application of Forward-to-word, step 4 (2) corresponds to an application of Backward-to-word, step 4 (1) (ii) and 4 (2) (ii) correspond to applications of Prefix, and *merge* corresponds to applications of Transitivity, Right-Congruence and Symmetry in $\mathcal{I}_c$. As a result, if $G \models \gamma(a, b)$, then by Word-to-forward in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists\, x\, (\alpha \cdot \beta(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(x,\, y))).$$

Similarly, if $G \models \gamma(b, a)$, then by Word-to-backward in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists\, x\, (\alpha \cdot \beta(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x))).$$

■

From Proposition 6.16, Algorithm 6.2 and Theorem 6.14, the corollary below follows immediately.

Corollary 6.17: In the context of $\mathcal{DM}$, the implication and finite implication problems for P_c are decidable in cubic-time. ■

6.3 The implication problems for P_c^-

This section shows that in contrast to Corollary 6.1, the implication and finite implication problems for P_c^- are decidable in the context of the deterministic data model.

Proposition 6.18: In the context of $\mathcal{DM}$, the implication and finite implication problems for P_c^- are decidable. ∎

Proof: To establish the decidability of the implication and finite implication problems for P_c^-, it suffices to give a finite model argument. That is, it suffices to show the following claim.

Claim: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c^-, and let $\phi = \bigwedge \Sigma \wedge \neg\varphi$. If there is a deterministic structure G such that $G \models \phi$, then there is a finite deterministic structure H such that $H \models \phi$.

For if the claim holds, then the implication and finite implication problems for P_c^- coincide and are decidable.

To show the claim, assume that there is a deterministic structure G satisfying ϕ. Recall that a constraint ψ of P_c^- is of either the form

- $\forall x\,(pf(\psi)(r,\, x) \rightarrow \forall y\,(lt(\psi)(x,\, y) \rightarrow rt(\psi)(x,\, y)))$ (i.e., the forward form), or the form
- $\forall x\,(pf(\psi)(r,\, x) \rightarrow \forall y\,(lt(\psi)(x,\, y) \rightarrow rt(\psi)(y,\, x)))$ (i.e., the backward form),

where $pf(\psi)$, $lt(\psi)$ and $rt(\psi)$ are $*$-free regular expressions, as described in Definition 3.2. Let

$$\begin{aligned} PEs(\phi) &= \{pf(\psi)\cdot lt(\psi),\ pf(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the forward form}\} \\ &\quad \cup\ \{pf(\psi)\cdot lt(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the backward form}\}, \\ Pts(\phi) &= \{\varrho \mid \varrho \text{ is a path},\ p \in PEs(\phi),\ \varrho \in p\}, \\ CloPts(\phi) &= \{\rho \mid \varrho \in Pts(\phi),\ \rho \preceq \varrho\}. \end{aligned}$$

Here $\varrho \in p$ means that path ϱ is in the regular language generated by $*$-free regular expression p, and $\rho \preceq \varrho$ stands for that path ρ is a prefix of path ϱ. Let E_ϕ be the set of edge labels appearing in some path in $Pts(\phi)$. Then we define H to be $(|H|,\, r^H,\, E^H)$ such that

- $|H| = \{a \mid a \in |G|,\ \rho \in CloPts(\phi),\ G \models \rho(r^G,\, a)\}$,

- $r^H = r^G$,
- for all $a, b \in |H|$ and $K \in E$, $H \models K(a, b)$ iff $K \in E_\phi$ and $G \models K(a, b)$.

It is easy to verify that $H \models \phi$ and H is deterministic, since G has these properties. By Lemma 6.3, the size of $|H|$ is at most the cardinality of $CloPts(\phi)$, which is finite because the regular language generated by a $*$-free regular expression is finite. This proves the claim. It should be noted that E_ϕ and $CloPts(\phi)$ are determined by ϕ only. ∎

6.4 The implication problems for P_c^*

This section establishes an undecidability result in the context of $\mathcal{DM}$.

Theorem 6.19: In the context of $\mathcal{DM}$, the implication and finite implication problems for P_c^* are undecidable. ∎

Theorem 6.19 shows that the determinism condition of $\mathcal{DM}$ does not reduce the analysis of path constraint implication to a trivial problem.

We prove Theorem 6.19 by reduction from the word problem for (finite) monoids. Before we present the details of the proof, we first review the word problem for (finite) monoids.

6.4.1 The word problem for (finite) monoids

Recall the following notions from [5, 62].

Definition 6.1: A *monoid* is a triple $(M, \circ, 1)$, where

- M is a nonempty set,
- $\circ$ is an associative binary relation on M, and
- 1 is an element of M that is the identity for $\circ$. That is, for any $a \in M$, $1 \circ a = a = a \circ 1$.

A monoid $(M, \circ, 1)$ is said to be finite if M is finite. ∎

Definition 6.2: Let , be a finite alphabet. The *free monoid generated by* , is $(, ^*, \cdot, \epsilon)$, where

- $, ^*$ is the set of all finite strings with letters in , ,
- $\cdot$ is the concatenation operator on strings, and
- ϵ is the empty string.

∎

Definition 6.3: Let , be a finite alphabet. An *equation* (over ,) is a pair (α, β) of strings in $, ^*$.

Let a finite set of equations

$$\Theta = \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in , ^*, \ i \in [1, n]\},$$

and a *test equation* θ be (α, β), where $\alpha, \beta \in , ^*$. Then $\Theta \models \theta$ ($\Theta \models_f \theta$) if for every (finite) monoid $(M, \circ, 1)$ and every homomorphism $h : , ^* \to M$, if $h(\alpha_i) = h(\beta_i)$ for each $i \in [1, n]$, then $h(\alpha) = h(\beta)$.

The *word problem for (finite) monoids* is the problem of determining, given Θ and θ, whether $\Theta \models \theta$ ($\Theta \models_f \theta$). ∎

The following result is well-known (see, e.g. [5, 62]).

Theorem 6.20: Both the word problem for monoids and the word problem for finite monoids are undecidable. ∎

6.4.2 The undecidability

We prove Theorem 6.19 by reduction from the word problem for (finite) monoids. To do this, we present an encoding of the word problem for (finite) monoids in terms of the (finite) implication problem for P_c^*.

Let $,_0$ be a finite alphabet and Θ_0 be a finite set of equations over $,_0$. Without loss of generality, assume $,_0 \subseteq E$, where E is the set of binary relation symbols in σ. Assume

$$\begin{aligned} ,_0 &= \{K_j \mid j \in [1,m],\ K_i \neq K_j \text{ if } i \neq j\}, \\ \Theta_0 &= \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in ,_0^*,\ i \in [1,n]\}. \end{aligned}$$

Note here that each symbol in $,_0$ is a binary relation symbol in E. Therefore, every α in $,_0^*$ can be represented as a path formula, also denoted by α. In addition, we use $\cdot$ to denote the concatenation operator for both paths and strings.

Let e_0 be the regular expression defined by:

$$e_0 = (K_1 + K_2 + \ldots + K_m)^*$$

We encode Θ_0 in terms of a subset Σ of P_c^*, as follows:

$$\begin{aligned} \Sigma &= \{\forall x\,(e_0(r,\,x) \rightarrow \forall y\,(\alpha_i(x,\,y) \rightarrow \beta_i(x,\,y))) \mid i \in [1,n]\} \ \cup \\ &\quad \{\forall x\,(e_0(r,\,x) \rightarrow \forall y\,(\beta_i(x,\,y) \rightarrow \alpha_i(x,\,y))) \mid i \in [1,n]\}. \end{aligned}$$

Let $(\alpha,\,\beta)$ be a test equation, where α and β are arbitrary strings in $,_0^*$. We encode this test equation as

$$\varphi = \forall x\,(e_0(r,\,x) \rightarrow \forall y\,(\alpha(x,\,y) \rightarrow \beta(x,\,y))).$$

It should be noted that in the encoding above, only forward constraints of P_c^* are used.

The lemma below shows that the encoding above is indeed a reduction from the word problem for (finite) monoids.

Lemma 6.21: Let Θ_0, $(\alpha,\,\beta)$, Σ and φ be given as above. Then in the context of $\mathcal{DM}$,

$$\Theta_0 \models (\alpha,\,\beta) \quad \text{iff} \quad \Sigma \models \varphi, \tag{a}$$

$$\Theta_0 \models_f (\alpha,\,\beta) \quad \text{iff} \quad \Sigma \models_f \varphi. \tag{b}$$

■

Proof: We prove (b) only. The proof of (a) is similar and simpler.

(*if*) Suppose that $\Theta_0 \not\models_f (\alpha, \beta)$. Then we show that $\Sigma \not\models_f \varphi$. That is, we show that there exists a finite deterministic structure G, such that $G \models \Sigma$ and $G \not\models \varphi$.

To do this, we first define some notations. By $\Theta_0 \not\models_f (\alpha, \beta)$, there exist a finite monoid $(M, \circ, 1)$ and a homomorphism $h : {}^{*}_{0} \to M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Based on M and h, we define an equivalence relation $\approx$ on ${}^{*}_{0}$ as follows:

$$\rho \approx \varrho \quad \text{iff} \quad h(\rho) = h(\varrho).$$

For every $\rho \in {}^{*}_{0}$, let $\widehat{\rho}$ be the equivalence class of ρ with respect to $\approx$. Let

$$C_{\Theta_0} = \{\widehat{\rho} \mid \rho \in {}^{*}_{0}\}.$$

Using these notations, we construct a deterministic structure $G = (|G|, r^G, E^G)$ as follows.

(1) $|G|$.

For each $\widehat{\rho} \in C_{\Theta_0}$, let $o(\widehat{\rho})$ be a distinct node. Then we define

$$|G| = \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}.$$

(2) $r^G = o(\widehat{\epsilon})$.

(3) The binary relations are populated as follows: For every $K \in E$ and $o(\widehat{\rho}), o(\widehat{\varrho}) \in |G|$, $G \models K(o(\widehat{\rho}), o(\widehat{\varrho}))$ iff $\rho \cdot K \in \widehat{\varrho}$.

Next, we show that G is indeed the structure desired. More specifically, we verify the following claims.

Claim 1: G is a finite deterministic structure.

To show that G is finite, it is sufficient to show that the set C_{Θ_0} is finite. Consider a function $f : C_{\Theta_0} \to M$ defined by

$$f : \ \widehat{\rho} \mapsto h(\rho).$$

Clearly, f is well-defined, total and injective. Therefore, because M is finite, C_{Θ_0} is also finite.

We next show that G is deterministic. By the construction of G, it is easy to see that for every $\rho \in {}^{*}_{0}$ and $j \in [1, m]$, $o(\widehat{\rho \cdot K_j})$ is the unique node such that $G \models K_j(o(\widehat{\rho}),\, o(\widehat{\rho \cdot K_j}))$. This is because h is a homomorphism, and as a result, if $\rho_1 \approx \rho_2$, then

$$\begin{aligned} h(\rho_1 \cdot K_j) &= h(\rho_1) \circ h(K_j) \\ &= h(\rho_2) \circ h(K_j) \\ &= h(\rho_2 \cdot K_j). \end{aligned}$$

Claim 2: $G \models \Sigma$.

Suppose, for *reductio*, that there is $i \in [1, n]$ such that

$$G \not\models \forall\, x\, (e_0(r,\, x) \rightarrow \forall\, y\, (\alpha_i(x,\, y) \rightarrow \beta_i(x,\, y))).$$

Then there is $\gamma \in e_o$ and $o(\widehat{\rho}), o(\widehat{\varrho}) \in |G|$ such that

$$G \models \gamma(r^G,\, o(\widehat{\rho})) \wedge \alpha_i(o(\widehat{\rho}),\, o(\widehat{\varrho})) \wedge \neg\beta_i(o(\widehat{\rho}),\, o(\widehat{\varrho})).$$

To see that this leads to a contradiction, it suffices to show:

Fact 1: For all $o(\widehat{\rho}), o(\widehat{\varrho}) \in |G|$ and $\xi \in {}^{*}_{0}$,

$$G \models \xi(o(\widehat{\rho}),\, o(\widehat{\varrho})) \;\text{ iff }\; \widehat{\rho \cdot \xi} = \widehat{\varrho}.$$

For if Fact 1 holds, then by the assumption, $\widehat{\rho \cdot \alpha_i} = \widehat{\varrho}$, but $\widehat{\rho \cdot \beta_i} \neq \widehat{\varrho}$. However, since h is a homomorphism and $\alpha_i \approx \beta_i$, we have

$$\begin{aligned} h(\rho \cdot \alpha_i) &= h(\rho) \circ h(\alpha_i) \\ &= h(\rho) \circ h(\beta_i) \\ &= h(\rho \cdot \beta_i). \end{aligned}$$

Thus $\rho \cdot \alpha_i \approx \rho \cdot \beta_i$. Hence $\widehat{\rho \cdot \alpha_i} = \widehat{\rho \cdot \beta_i}$. This contradicts the assumption.

Similarly, we can also show that for every $i \in [1, n]$,

$$G \models \forall\, x\, (e_0(r,\, x) \rightarrow \forall\, y\, (\beta_i(x,\, y) \rightarrow \alpha_i(x,\, y))).$$

We show Fact 1 by induction on $|\xi|$.

Base case: $\xi = \epsilon$.

Clearly, $G \models \epsilon(o(\widehat{\rho}),\, o(\widehat{\varrho}))$ iff $o(\widehat{\rho}) = o(\widehat{\varrho})$ iff $\widehat{\rho} = \widehat{\varrho}$.

Inductive step: Assume Fact 1 for ξ. We next show that Fact 1 also holds for $\xi \cdot K$.

If $G \models \xi \cdot K(o(\widehat{\rho}),\, o(\widehat{\varrho}))$, then by the induction hypothesis and Lemma 6.3, we have

$$G \models \xi(o(\widehat{\rho}),\, o(\widehat{\rho \cdot \xi})) \wedge K(o(\widehat{\rho \cdot \xi}),\, o(\widehat{\varrho})).$$

By the definition of G, $G \models K(o(\widehat{\rho \cdot \xi}),\, o(\widehat{\varrho}))$ iff $\rho \cdot \xi \cdot K \in \widehat{\varrho}$. Therefore,

$$\rho \cdot \widehat{\xi \cdot K} = \widehat{\varrho}.$$

Conversely, suppose that $\rho \cdot \widehat{\xi \cdot K} = \widehat{\varrho}$. By the induction hypothesis,

$$G \models \xi(o(\widehat{\rho}),\, o(\widehat{\rho \cdot \xi})).$$

Again by the definition of G, $G \models K(o(\widehat{\rho \cdot \xi}),\, o(\rho \cdot \widehat{\xi \cdot K}))$. Hence by $\rho \cdot \widehat{\xi \cdot K} = \widehat{\varrho}$, we have

$$G \models \xi \cdot K(o(\widehat{\rho}),\, o(\widehat{\varrho})).$$

Claim 3: $G \not\models \varphi$.

By Fact 1, $G \models \alpha(r^G,\, o(\widehat{\alpha}))$. By $h(\alpha) \neq h(\beta)$, we have

$$o(\widehat{\alpha}) \neq o(\widehat{\beta}).$$

Therefore, again by Fact 1, we have $G \not\models \beta(r^G,\, o(\widehat{\alpha}))$. Hence

$$G \models \alpha(r^G,\, o(\widehat{\alpha})) \wedge \neg\beta(r^G,\, o(\widehat{\alpha})).$$

Note that $\epsilon \in e_0$. That is, the empty path ϵ is in the language generated by the regular expression e_0. Thus

$$G \models \exists\, x\, y\, (e_0(r^G,\, x) \wedge \alpha(x,\, y) \wedge \neg\beta(x,\, y)).$$

That is, $G \not\models \varphi$.

(*only if*) Suppose that there exists a finite deterministic structure G such that $G \models \Sigma$ and $G \not\models \varphi$. Then we show that $\Theta_0 \not\models_f (\alpha, \beta)$. More specifically, we define a finite monoid $(M, \circ, 1)$ and a homomorphism $h : , \overset{*}{0} \to M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation $\sim$ on $, \overset{*}{0}$, as follows:

$$\begin{aligned} \rho \sim \varrho \text{ iff } G \models & \forall x(e_0(r, x) \to \forall y\,(\rho(x, y) \to \varrho(x, y))) \wedge \\ & \forall x\,(e_0(r, x) \to \forall y\,(\varrho(x, y) \to \rho(x, y))). \end{aligned}$$

Then by $G \models \Sigma$, for every $i \in [1, n]$, we have $\alpha_i \sim \beta_i$. In addition, by $G \not\models \varphi$, we have $\alpha \not\sim \beta$.

For every $\rho \in , \overset{*}{0}$, let $[\rho]$ denote the equivalence class of ρ with respect to $\sim$. Then clearly, for every $i \in [1, n]$, $[\alpha_i] = [\beta_i]$. However, we have $[\alpha] \neq [\beta]$.

Using the notion of $\sim$, we define

$$M = \{[\rho] \mid \rho \in , \overset{*}{0}\}.$$

An important property of M is described as follows.

Claim 4: M is finite.

To show this, for every $\rho \in , \overset{*}{0}$, let

$$S_\rho = \{(a, b) \mid a, b \in |G|,\ G \models e_0(r^G, a) \wedge \rho(a, b)\}.$$

In addition, let

$$S_G = \{S_\rho \mid \rho \in , \overset{*}{0}\}.$$

Since $S_\rho \subseteq |G| \times |G|$ and $|G|$ is finite, S_G is finite. Moreover, it is easy to verify the following:

Fact 2: For all $\rho, \varrho \in , \overset{*}{0}$, $\rho \sim \varrho$ iff $S_\rho = S_\varrho$.

To see that Fact 2 holds, first assume that $\rho \sim \varrho$. Then for each $(a, b) \in S_\rho$, by the definition of S_ρ, we have

$$G \models e_0(r^G, a) \wedge \rho(a, b).$$

By the definition of $\sim$ and the assumption that $\rho \sim \varrho$, we have

$$G \models e_0(r^G,\, a) \wedge \varrho(a,\, b).$$

Hence $(a,\, b) \in S_\varrho$. Therefore, $S_\rho \subseteq S_\varrho$. Similarly, it can be shown that $S_\varrho \subseteq S_\rho$. Hence

$$S_\rho = S_\varrho.$$

Conversely, assume that $S_\rho = S_\varrho$. Suppose, for *reductio*, that $\rho \not\sim \varrho$. Without loss of generality, assume that

$$G \not\models \forall\, x(e_0(r, x) \rightarrow \forall\, y\,(\rho(x, y) \rightarrow \varrho(x, y))).$$

Then there exist $a, b \in |G|$, such that

$$G \models e_0(r^G,\, a) \wedge \rho(a,\, b) \wedge \neg\varrho(a,\, b).$$

That is, $(a,\, b) \in S_\rho$ but $(a,\, b) \notin S_\varrho$. Hence $S_\rho \neq S_\varrho$. This contradicts the assumption. Therefore, Fact 2 holds.

Next, consider a function $g : M \rightarrow S_G$ defined by

$$g :\ [\rho] \mapsto S_\rho.$$

Using Fact 2 above, it is easy to see that g is well-defined, total and injective. Therefore, because S_G is finite, M is also finite.

Next, we define a binary operation $\circ$ on M by

$$[\rho] \circ [\varrho] = [\rho \cdot \varrho].$$

It is easy to verify the following claims.

Claim 5: $\circ$ is well-defined.

To see this, for all $\rho_1, \rho_2, \varrho_1, \varrho_2 \in \,,^*_0$ such that $\rho_1 \sim \rho_2$ and $\varrho_1 \sim \varrho_2$, we show that

$$\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2.$$

To do this, consider all $o, o_1 \in |G|$ such that

$$G \models e_0(r^G,\, o) \wedge \rho_1 \cdot \varrho_1(o,\, o_1).$$

Clearly, there exists $o' \in |G|$ such that

$$G \models \rho_1(o,\, o') \wedge \varrho_1(o',\, o_1).$$

By $\rho_1 \sim \rho_2$, we have

$$G \models \rho_2(o,\, o').$$

Note that $e_0 \cdot \rho_2 \subseteq e_0$. That is, the language generated by the regular expression $e_0 \cdot \rho$ is contained in the language generated by e_0. By $\varrho_1 \sim \varrho_2$, we also have

$$G \models \varrho_2(o',\, o_1).$$

Hence

$$G \models \rho_2 \cdot \varrho_2(o,\, o_1).$$

Therefore,

$$G \models \forall\, x\, (e_0(r, x) \rightarrow \forall\, y\, (\rho_1 \cdot \varrho_1(x,\, y) \rightarrow \rho_2 \cdot \varrho_2(x, y))).$$

Similarly, we can show that

$$G \models \forall\, x\, (e_0(r, x) \rightarrow \forall\, y\, (\rho_2 \cdot \varrho_2(x, y) \rightarrow \rho_1 \cdot \varrho_1(x,\, y))).$$

Therefore, $\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2$. Hence $\circ$ is well-defined.

Claim 6: $\circ$ is associative.

This is because for all $[\rho], [\varrho], [\xi] \in M$,

$$\begin{aligned}
([\rho] \circ [\varrho]) \circ [\xi] &= [\rho \cdot \varrho] \circ [\xi] \\
&= [\rho \cdot \varrho \cdot \xi] \\
&= [\rho] \circ ([\varrho \cdot \xi]) \\
&= [\rho] \circ ([\varrho] \circ [\xi]).
\end{aligned}$$

Claim 7: $[\epsilon]$ is the identity for $\circ$. This is because for any $[\rho] \in M$,

$$[\epsilon] \circ [\rho] = [\rho] = [\rho] \circ [\epsilon].$$

These claims show that $(M, \circ, [\epsilon])$ is a finite monoid.

Finally, we define $h : , {}^{*}_{0} \to M$ by

$$h : \ \rho \mapsto [\rho].$$

Clearly, h is a homomorphism since

$$h(\rho \cdot \varrho) = [\rho \cdot \varrho] = [\rho] \circ [\varrho] = h(\rho) \circ h(\varrho).$$

In addition, for every $i \in [1, n]$, by $[\alpha_i] = [\beta_i]$, $h(\alpha_i) = h(\beta_i)$. Moreover, by $[\alpha] \neq [\beta]$, $h(\alpha) \neq h(\beta)$. Therefore,

$$\Theta_0 \not\models_f (\alpha, \beta).$$

This completes the proof of Lemma 6.21. ∎

From Lemma 6.21 and Theorem 6.20, Theorem 6.19 follows immediately.

Part III

Path Constraints on Structured Data

In Part II, a number of results on path constraint implication have been established in the context of semistructured databases, i.e., databases without schemas. From these results we have developed a reasonable understanding – in the context of untyped data – of the interesting decision problems for such constraints. There are useful restrictions of path constraints with a decidable implication problem. One might be tempted to think that the imposition of a type system, which imposes some regularity on the data, would be to generate new classes of path constraints with decidable implication problems. This may be the case. However one of the main results of this work is to establish the possibly surprising result that the presence of types actually *complicates* the implication problem for path constraints: there are decidable path constraint problems that become undecidable in the presence of types. Moreover the type used in the construction of this result is not particularly "pathological".

In Part III, we consider path constraint implication in the context of structured databases, i.e., databases constrained by a type system or schema.

What is the difference between path constraint implication in the context of semistructured data as opposed to structured data? In structured databases, path constraint implication is considered in connection with a schema. More specifically, the implication problem for path constraints over a schema Δ is the problem of determining, given a finite set $\Sigma \cup \{\varphi\}$ of path constraints, whether all the database instances of Δ that satisfy Σ are also models of φ. Here an instance of the schema Δ has a certain structure specified by Δ. In other words, an instance of Δ must satisfy certain type constraints imposed by Δ. In contrast, a semistructured database is free of type constraints.

Here we address the question whether there is interaction between type constraints and

path constraints. We show that some results on path constraint implication in semistructured databases no longer hold in the presence of types. For example, consider the implication and finite implication problems for the path constraint language P_c. In the context of the semistructured data model $\mathcal{SM}$, as shown in Chapter 4, these implication problems are undecidable. In the typed context, however, the (finite) implication problem for P_c over a schema is decidable as long as the schema does not contain recursive types, i.e., self-referential data structures. This is because in any instance of such a schema, there are only finitely many navigation paths. In other words, the language P_c over the schema has only finitely many sentences up to equivalence, and therefore, its associated implication problem is decidable. In addition, schemas with recursive types may also have severe impact. In Part III, we show that these schemas may in some cases simplify the analysis of path constraint implication, and in other cases make it harder. More specifically, we show that on the one hand, there are implication problems associated with path constraints that are decidable in the context of semistructured data but that become undecidable in the presence of these schemas. On the other hand, the reverse can also occur.

One may wonder why imposing a schema on the data can alter the computational complexity of the path constraint implication problem in unexpected ways. For orientation, we provide intuitive background here. An implication problem for a logical language L is determined by a collection of structures $\mathcal{S}$ which interpret that language. We say that a finite set Σ of L sentences $\mathcal{S}$-*implies* an L sentence φ just in case for every structure $G \in \mathcal{S}$, if $G \models \Sigma$, then $G \models \varphi$. Suppose we are given two classes of structures $\mathcal{S}' \subset \mathcal{S}$, each interpreting L. In general, the computational complexity of the $\mathcal{S}$-implication problem for L may bear no obvious connection to the complexity of the $\mathcal{S}'$-implication problem for L. A justly famous example of this is given by the case where L is the collection of all first-order sentences with a single binary relation and $\mathcal{S}$ and $\mathcal{S}'$ are the classes of all relational structures and all finite relational structures respectively. Then, the completeness theorem for first-order logic and Church's Theorem together tell us that the $\mathcal{S}$-implication problem for L is r.e.-complete, while Trahktenbrot's Theorem tells us that the $\mathcal{S}'$-implication problem for L is co-r.e.-complete (see, e.g., [15]). Note that in this example, $\mathcal{S}'$ is not first order definable over $\mathcal{S}$.

In Part III, we will study implication problems for collections of path constraints which can be represented as proper fragments L^* of first-order logic. Again, let $\mathcal{S}$ be the collection of all structures. When we consider the $\mathcal{S}$-implication problem for L^* in the context of a type constraint Φ, what we really mean is the $\mathcal{S}''$-implication problem for L^* where $\mathcal{S}''$ is the collection of structures in $\mathcal{S}$ which satisfy the type constraint Φ. In Part III, we will give examples where the $\mathcal{S}$-implication problem for L^* is undecidable, but the $\mathcal{S}''$-implication problem for L^* is decidable. This sort of situation is quite familiar. For example, the $\mathcal{S}$-implication problem for first-order logic is undecidable, but the $\mathcal{S}''$-implication problem for first-order logic is decidable when $\mathcal{S}''$ is the collection of linear orderings (and this collection *is* determined by a first order "constraint"). On the other hand, also in Part III, we exhibit situations in which the $\mathcal{S}$-implication problem for L^* is decidable, but the $\mathcal{S}''$-implication problem for L^* is undecidable. This possibility is perhaps a bit less familiar, namely the possibility that by imposing a restriction on a collection of structures we can turn a decidable implication problem into an undecidable implication problem. Indeed, in the context where L is the collection of all first-order sentences and the restriction itself is first order, this is clearly *impossible*, since in this case, the implication problem for the restricted class is simply a special case of the unrestricted implication problem. But in the context of the interaction between path and type constraints, this is precisely not the case. Namely, the type constraints we consider *cannot* be expressed in the path constraint languages in question. We hope this observation will clarify the results of Part III, which exhibit a path constraint implication problem which is decidable with respect to a collection of structures $\mathcal{S}$, but is undecidable with respect to the collection of structures $G \in \mathcal{S}$ which satisfy a given type constraint Φ.

To disclose the interaction between type constraints and path constraints, we study path constraint implication in the context of three practical object-oriented data models: $\mathcal{M}$, $\mathcal{M}^+$ and $\mathcal{M}^+_f$. These models are similar to those considered in [3, 5, 6, 31, 58]. They support classes, the record and (finite) set constructs, and recursive structures. However, like the model studied in [3], these models do not support complex value equality (a notion to be addressed in Chapter 10).

These object-oriented data models are formally defined in Chapter 7. An abstraction of

databases is given in terms of type constraints for each of these models. In addition, the definitions of path constraints and path constraint implication are also refined in the context of these models.

In Chapter 8, the implication and finite implication problems for P_c are investigated in the context of these object-oriented data models. We show that these problems are decidable in cubic-time in $\mathcal{M}$, but are undecidable in $\mathcal{M}^+$ and $\mathcal{M}^+_f$. However, in $\mathcal{M}^+$ and $\mathcal{M}^+_f$, we show that the implication and finite implication problems for the class of word constraints, P_w, are decidable.

A full treatment of the interaction between path and type constraints is presented in Chapter 9. We show that on the one hand, there are implication problems associated with path constraints that are undecidable in the semistructured data model $\mathcal{SM}$, but that become decidable in cubic-time in the object-oriented model $\mathcal{M}$. On the other hand, there are also implication problems that are decidable in PTIME in $\mathcal{SM}$, but that become undecidable in $\mathcal{M}^+$ and $\mathcal{M}^+_f$.

Finally, we investigate the impact of complex value equality on path constraint implication in Chapter 10. To do this, we introduce another object-oriented model, $\mathcal{M}^{\approx}$, which supports complex values with nested structures. We show that complex value equality can be characterized in terms of equality constraints, and equality constraints also interact with path constraints. In the context of $\mathcal{M}^{\approx}$, we show that the implication and finite implication problems for the class of word constraints, P_w, remain decidable.

Chapter 7

Object-Oriented Data Models

This chapter introduces three object-oriented data models: $\mathcal{M}_f^+$, $\mathcal{M}^+$ and $\mathcal{M}$, presents an abstraction of databases for each of these models in terms of type constraints, and refines the definitions of path constraints and path constraint implication in the context of these models.

We begin with a presentation of the model $\mathcal{M}_f^+$ (Section 7.1). Similar to the models studied in [3, 5, 6, 31, 58], $\mathcal{M}_f^+$ supports classes, the record and finite set constructs, and recursive structures. We then describe the model $\mathcal{M}^+$ (Section 7.2), which is same as $\mathcal{M}_f^+$ except that it supports the set construct instead of the finite set construct. That is, sets in $\mathcal{M}^+$ are not necessarily finite. Finally, we study the model $\mathcal{M}$ (Section 7.3), which supports classes, the record construct, and recursive structures, but does not allow sets or finite sets.

7.1 The model $\mathcal{M}_f^+$

We first define database schemas and their instances in $\mathcal{M}_f^+$, and then present an abstraction of databases in $\mathcal{M}_f^+$ in terms of type constraints. Finally, we refine the definitions of path constraints and path constraint implication in the context of $\mathcal{M}_f^+$.

7.1.1 Database schemas and instances

Assume a fixed countable set of labels, $\mathcal{L}$, and a fixed finite set of *base types*, $\mathcal{B}$. Examples of base types include *int* and *string*.

Definition 7.1: Let $\mathcal{C}$ be some finite set of *classes*. The set of *types over* $\mathcal{C}$, $Types^{\mathcal{C}}$, is defined by the syntax:

$$\begin{aligned} t &::= b \mid C \\ \tau &::= t \mid \{t\} \mid [l_1 : t_1, \ldots, l_n : t_n] \end{aligned}$$

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$. The notations $\{t\}$ and $[l_1 : t_1, \ldots, l_n : t_n]$ represent *set type* and *record type*, respectively. We reserve τ to range over $Types^{\mathcal{C}}$. ■

Definition 7.2: A *schema* is a triple $\Delta = (\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ is a finite set of classes,
- ν is a mapping: $\mathcal{C} \rightarrow Types^{\mathcal{C}}$ such that for each $C \in \mathcal{C}$, $\nu(C) \notin \mathcal{B} \cup \mathcal{C}$, and
- $DBtype \in Types^{\mathcal{C}} \setminus (\mathcal{B} \cup \mathcal{C})$.

■

Here we assume that every database of a schema has a unique (persistent) entry point, and $DBtype$ in the schema specifies the type of the entry point.

Example 7.1: An example schema of $\mathcal{M}_f^+$ is $\Delta_0 = (\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ consists of a single class $Person$,
- ν maps $Person$ to a record type $[name : string, spouse : Person]$, and
- $DBtype$ is $\{Person\}$.

■

Definition 7.3: A *database instance* of schema $(\mathcal{C}, \nu, DBtype)$ is a triple $I = (\pi, \mu, d)$, where

- π is an *oid* (object identity) *assignment* that maps each $C \in \mathcal{C}$ to a finite set of oids, $\pi(C)$, such that for all $C, C' \in \mathcal{C}$, $\pi(C) \cap \pi(C') = \emptyset$ if $C \neq C'$;

- for each $C \in \mathcal{C}$, μ maps each oid in $\pi(C)$ to a value in $[\![\nu(C)]\!]_\pi$, where

$$\begin{aligned}
[\![b]\!]_\pi &= D_b, \\
[\![C]\!]_\pi &= \pi(C), \\
[\![\{\tau\}]\!]_\pi &= \{V \mid V \subseteq [\![\tau]\!]_\pi,\ V \text{ is finite}\}, \\
[\![[l_1 : \tau_1, ..., l_n : \tau_n]]\!]_\pi &= \{[l_1 : v_1, ..., l_n : v_n] \mid v_i \in [\![\tau_i]\!]_\pi,\ i \in [1, n]\};
\end{aligned}$$

  here D_b denotes the domain of base type b;

- d is a value in $[\![DBtype]\!]_\pi$, which represents the (persistent) entry point into the database instance.

We denote the set of all database instances of schema Δ by $\mathcal{I}(\Delta)$. ■

Example 7.2: An instance of the schema Δ_0 given in Example 7.1 is (π, μ, d), where

- $\pi(Person) = \{p_1, p_2, p_3, p_4\}$,
- $\mu : \pi(Person) \rightarrow [\![[name : string, spouse : Person]]\!]_\pi$ is defined by:

$$\begin{aligned}
\mu(p_1) &\mapsto [name : \text{“Smith”}, spouse : p_2] \\
\mu(p_2) &\mapsto [name : \text{“Mary”}, spouse : p_1] \\
\mu(p_3) &\mapsto [name : \text{“Joe”}, spouse : p_4] \\
\mu(p_4) &\mapsto [name : \text{“Maria”}, spouse : p_3]
\end{aligned}$$

- $d = \{p_1, p_2, p_3, p_4\}$.

■

7.1.2 Abstraction of databases

We next present an abstraction of databases in $\mathcal{M}_f^+$. Since structured data can be viewed as semistructured data further constrained by a schema, along the same lines of the abstraction of semistructured databases described in Chapter 3, we represent a structured

database as a first-order logic structure satisfying a certain type constraint determined by its schema. Such a structure can also be depicted as an edge-labeled, rooted, directed graph.

To do this, we first define the first-order signature determined by a schema. Two components of such signatures are described as follows.

Definition 7.4: Given a schema $\Delta = (\mathcal{C}, \nu, DBtype)$, we define *the set of binary relation symbols* and *the set of types determined by* Δ, denoted by $E(\Delta)$ and $T(\Delta)$, respectively, to be the smallest sets having the following properties:

- $DBtype \in T(\Delta)$ and $\mathcal{C} \subseteq T(\Delta)$;
- if $DBtype = \{\tau\}$ (or for some $C \in \mathcal{C}$, $\nu(C) = \{\tau\}$), then τ is in $T(\Delta)$ and $*$ is in $E(\Delta)$;
- if $DBtype = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or for some $C \in \mathcal{C}$, $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, τ_i is in $T(\Delta)$ and l_i is in $E(\Delta)$.

■

Note here we use the distinguished binary relation $*$ to denote the set membership relation. This differs slightly from the presentation in Chapter 1.

Obviously, both $E(\Delta)$ and $T(\Delta)$ are finite. In addition, every type in $T(\Delta)$ except $DBtype$ is either a class type or a base type. That is,

$$T(\Delta) \subseteq \mathcal{C} \cup \mathcal{B} \cup \{DBtype\}.$$

Definition 7.5: The *signature determined by schema* Δ, $\sigma(\Delta)$, is a triple

$$(r, E(\Delta), R(\Delta)),$$

where r is a constant symbol (denoting the root), $E(\Delta)$ is the finite set of binary relation symbols (denoting the edge labels) defined above, and $R(\Delta)$ is the finite set of unary relation symbols (denoting the sorts) defined by $\{R_\tau \mid \tau \in T(\Delta)\}$. ■

For example, the signature determined by the schema given in Example 7.1 is (r, E, R), where

- r is a constant, which in each instance (π, μ, d) of the schema intends to name d;
- $E = \{*, name, spouse\}$; and
- $R = \{R_{DBtype}, R_{Person}, R_{string}\}$.

We specify a $\sigma(\Delta)$-structure G by giving $(|G|, r^G, E^G, R^G)$, where $|G|$, r^G and E^G are interpreted in the same way as in Chapter 2, and R^G is a set of unary relations on $|G|$. Each of these unary relations is named by a relation symbol in $R(\Delta)$. For each $R_\tau \in R(\Delta)$, we write R_τ^G for the relation in G named by R_τ.

We intend to represent an instance I of a schema Δ as a (finite) $\sigma(\Delta)$-structure G that satisfies a certain type constraint imposed by Δ. More specifically, let $\Delta = (\mathcal{C}, \nu, DBtype)$, $I = (\pi, \mu, d)$ and $G = (|G|, r^G, E^G, R^G)$. Intuitively, we use $|G|$, r^G, E^G and R^G to represent data entities, the entry point d, record labels and set membership, and types of the data entities, respectively. This representation must satisfy the type constraint imposed by Δ. Informally, the type constraint is stated as follows.

- Every element of $|G|$ must have a unique type in $T(\Delta)$. In particular, r^G is of $DBtype$.
- If an element a of $|G|$ has type τ, then a must satisfy the constraint imposed by τ:
  - If τ is a base type b, then a has no outgoing edge.
  - If $\tau = \{\tau'\}$, or τ is a class type C and $\nu(C) = \{\tau'\}$, then all the outgoing edges of a are labeled with $*$ and lead to elements of type τ'.
  - If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or τ is a class type C and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then a has exactly n outgoing edges. These edges are labeled with l_1, ..., l_n, respectively. In addition, for each $i \in [1, n]$, if $G \models l_i(a, b)$ for some $b \in |G|$, then b has type τ_i.

Formally, the type constraint imposed by a schema can be formulated as a sentence in two-variable logic with counting [15, 46], C^2. Two-variable logic FO^2 is the fragment of first-order logic consisting of all relational sentences with at most two distinct variables

[15, 45], and C^2 is the extension of FO^2 with counting quantifiers. In particular, below we use the counting quantifier $\exists !$, whose semantics is described as follows: structure G satisfies $\exists !x\, \psi(x)$ if and only if there exists a unique element a of G such that $G \models \psi(a)$.

Definition 7.6: Let Δ be a schema. For each τ in $T(\Delta)$, the *constraint determined by* τ is the sentence $\forall\, x\, \phi_\tau(x)$ defined as follows:

- if $\tau = b$, then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta)} \neg l(x, y));$$
- if for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = \{\tau'\}$ (or $\tau = DBtype = \{\tau'\}$), then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta) \setminus \{*\}} \neg l(x, y)) \ \wedge \forall\, y\, (*(x, y) \rightarrow R_{\tau'}(y));$$
- if for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $\tau = DBtype$ and $DBtype = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta) \setminus \{l_1, \ldots, l_n\}} \neg l(x, y)) \ \wedge \bigwedge_{i \in [1,n]} (\exists !\, y\, l_i(x, y) \ \wedge \forall\, y\, (l_i(x, y) \rightarrow R_{\tau_i}(y))).$$

The *type constraint determined by schema* Δ is the sentence $\Phi(\Delta)$ defined by:

$$R_{DBtype}(r) \wedge \bigwedge_{\tau \in T(\Delta)} \forall\, x\, \phi_\tau(x) \wedge \forall\, x\, (\bigvee_{\tau \in T(\Delta)} R_\tau(x) \wedge \bigwedge_{\tau \in T(\Delta)} (R_\tau(x) \rightarrow \bigwedge_{\tau' \in T(\Delta) \setminus \{\tau\}} \neg R_{\tau'}(x)))$$

■

Note here for simplicity, we assume that for each base type $b \in \mathcal{B}$, the domain of b, D_b, is infinite. If D_b is finite, i.e., the cardinality of D_b is some natural number n, then we define the constraint determined by b to be the following sentence in C^2:

$$\forall\, x\, \phi_b(x) \ \wedge \ \exists^{=n}\, x\, R_b(x).$$

Here $\phi_b(x)$ is the formula determined by b given in Definition 7.6, and $\exists^{=n}$ is another counting quantifier. The semantics of $\exists^{=n}$ is described as follows: a structure satisfies $\exists^{=n}\, x\, \psi(x)$ if and only if there are exactly n elements in the structure satisfying ψ. When base type b has a finite domain, we substitute this constraint for $\forall\, x\, \phi_b(x)$ in $\Phi(\Delta)$.

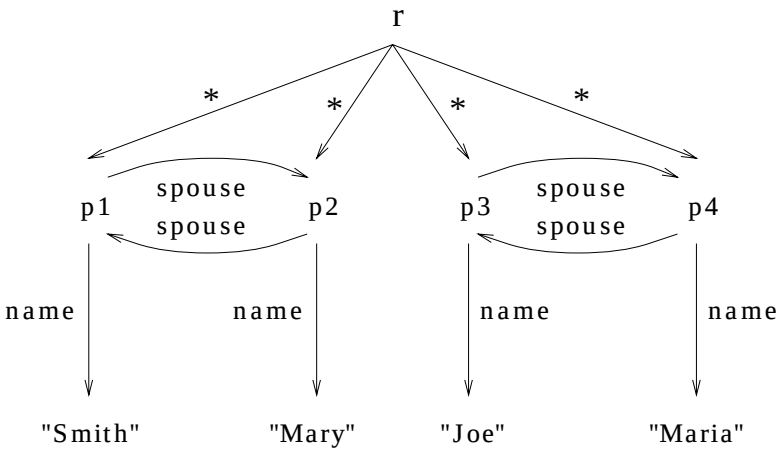

Figure 7.1: An example of abstract databases in $\mathcal{M}_f^+$

Using the type constraint defined above, we present an abstraction of databases in the object-oriented model as follows. Its justification will be given later in this section.

Definition 7.7: An *abstract database of a schema* Δ is a finite structure G of the signature $\sigma(\Delta)$ such that $G \models \Phi(\Delta)$.

We denote the set of all abstract databases of a schema Δ by $\mathcal{U}_f(\Delta)$. ∎

We use $\mathcal{U}(\Delta)$ to denote the set of all the structures of signature $\sigma(\Delta)$ satisfying the following conditions: for each $G \in \mathcal{U}(\Delta)$, $G \models \Phi(\Delta)$ and G respects *the finite set rule*. That is, for each set type $\tau \in T(\Delta)$ and each $o \in R_\tau^G$, there are only finitely many o' in G such that $G \models *(o, o')$. As a result, each node in G has finitely many outgoing edges.

An example structure is depicted in Figure 7.1. This structure represents the database instance given in Example 7.2.

7.1.3 Path constraints revisited

Next, we refine the definitions of paths and path constraints in the context of $\mathcal{M}_f^+$, and justify the abstraction of databases given above by considering path constraint satisfiability.

Why do we need to refine these definitions? Given the signature $(r, E(\Delta), R(\Delta))$ determined by a schema Δ, one could define paths and path constraints in the same way as in

Definitions 2.1 and 2.2. These definitions, however, are somewhat too coarse in the context of the object-oriented model. Because of the type constraint $\Phi(\Delta)$, some paths are not meaningful in the structures of $\mathcal{U}(\Delta)$. That is, there exists path $\alpha(x, y)$ defined in this way such that for all $G \in \mathcal{U}(\Delta)$ and all $a, b \in |G|$,

$$G \not\models \alpha(a, b).$$

Such paths are said to be *undefined over* Δ. A path constraint is said to be *trivial over schema* Δ if it is satisfied by either all the structures in $\mathcal{U}(\Delta)$ or by none of them. Path constraints containing undefined paths are trivial.

As an example, consider the schema Δ_0 given in Example 7.1. It is easy to verify that the path

$$\alpha_0(r, x) = \exists\, y\, (spouse(r, y) \wedge spouse(y, x)).$$

is undefined over Δ_0 by examining $\Phi(\Delta_0)$. As a result, the constraint

$$\forall\, x\, (\alpha_0(r, x) \rightarrow \beta(r, x))$$

is satisfied by all the structures in $\mathcal{U}(\Delta_0)$, and the constraint

$$\forall\, x\, (\beta(r, x) \rightarrow \alpha_0(r, x))$$

is not satisfied by any structure in $\mathcal{U}(\Delta_0)$ as long as $\beta(r, x)$ is not undefined over Δ_0.

Path constraints may also be trivial even if they do not contain undefined paths. For example, consider

$$\forall\, x\, (*(r, x) \rightarrow \exists\, y\, (*(r, y) \wedge name(y, x))).$$

It is easy to verify that this constraint cannot be satisfied by any structure G in $\mathcal{U}(\Delta_0)$. This is because $G \models \Phi(\Delta_0)$. As a result, for every $a \in |G|$ such that $G \models *(r^G, a)$, a has type $Person$, and for any $b \in |G|$ such that $G \models * \cdot name(r^G, b)$, b has type $string$. Again by $G \models \Phi(\Delta_0)$, every element of $|G|$ has a unique type. Hence this constraint does not hold in G.

We are not interested in trivial constraints since their satisfiability can be simply determined by examining the schema. These considerations suggest that we refine the definitions of paths and path constraints in the context of $\mathcal{M}_f^+$ to exclude trivial constraints.

Formally, we define paths in $\mathcal{M}_f^+$ in terms of finite state automata [51].

Definition 7.8: Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}_f^+$. The *finite state automata determined by* Δ is defined to be $M(\Delta) = (S, A, \delta, s, F)$, where

- the set of *states* S is $T(\Delta)$, the set of types determined by Δ;
- the *alphabet* A is $E(\Delta)$, the set of binary relation symbols determined by Δ;
- the *initial state* s is $DBtype$;
- the set of *final states* F is also $T(\Delta)$; and
- the *transition function* δ is defined as follows: For every $\tau \in T(\Delta)$,
  - if $\tau = \{\tau'\}$, or $\tau = C$ and $\nu(C) = \{\tau'\}$, then $\delta(\tau, *) = \tau'$;
  - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ for some $C \in \mathcal{C}$, then for every $i \in [1, n]$, $\delta(\tau, l_i) = \tau_i$.

Define a (partial) function $\widehat{\delta} : T(\Delta) \times E(\Delta)^* \to T(\Delta)$ by:

$$\begin{aligned} \widehat{\delta}(\tau, \epsilon) &= \tau \\ \widehat{\delta}(\tau, \alpha K) &= \delta(\widehat{\delta}(\tau, \alpha), K) \end{aligned}$$

Here $E(\Delta)^*$ stands for the Kleene closure of $E(\Delta)$.

A *path* α *over* Δ is an element of $E(\Delta)^*$ such that there is $\tau \in T(\Delta)$ and $\widehat{\delta}(\tau, \alpha)$ is defined.

Let $Paths(\Delta)$ be the language accepted by $M(\Delta)$. A *valid path over* Δ is an element of $Paths(\Delta)$. The *type of a valid path* α, $type(\alpha)$, is defined to be $\widehat{\delta}(DBtype, \alpha)$. ∎

It is easy to verify that $\widehat{\delta}$ is indeed a function. As a result, the type of a valid path is well-defined and unique. In particular, the empty path ϵ is in $Paths(\Delta)$ and $type(\epsilon) = DBtype$.

Example 7.3: The finite state automata determined by the schema Δ_0 given in Example 7.1 is shown in Figure 7.2. ∎

Equivalently, $Paths(\Delta)$ can be defined inductively as follows:

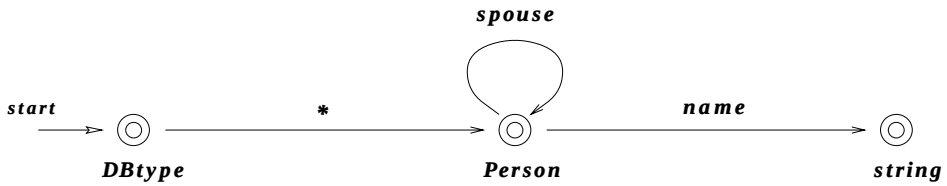

Figure 7.2: The finite state automata determined by the schema given in Example 7.1

- the empty path ϵ is in $Paths(\Delta)$ and $type(\epsilon) = DBtype$;
- for any $\alpha \in Paths(\Delta)$, where $type(\alpha) = \tau$,
  - if for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = \{\tau'\}$ (or $\tau = DBtype = \{\tau'\}$), then $\alpha \cdot *$ is a path in $Paths(\Delta)$ and $type(\alpha \cdot *) = \tau'$;
  - if there exists $C \in \mathcal{C}$ such that $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $\tau = DBtype = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, $\alpha \cdot l_i$ is in $Paths(\Delta)$ and $type(\alpha \cdot l_i) = \tau_i$.

As in Chapter 2, a path over a schema can be expressed as a first-order logic formula $\alpha(x, y)$, where x and y denote the tail and head nodes of the path, respectively. In the same way, the notions of path concatenation, prefix, suffix, and length can be defined.

It is easy to verify that every path over schema Δ is a suffix of some path in $Paths(\Delta)$. In addition, for every $\alpha \in Paths(\Delta)$ and every $\rho \preceq_p \alpha$, ρ is in $Paths(\Delta)$. That is, $Paths(\Delta)$ is prefix-closed. Here $\rho \preceq_p \alpha$ denotes that ρ is a prefix of α.

Next, we define path constraints in the context of $\mathcal{M}_f^+$.

Definition 7.9: A *path constraint* φ *over schema* Δ is an expression of either the *forward* form

$$\forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(x,\, y))),$$

or the *backward* form

$$\forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x))),$$

where α, β, γ are paths over Δ. In addition,

- if φ is of the forward form, then $\alpha \cdot \beta \in Paths(\Delta)$, $\alpha \cdot \gamma \in Paths(\Delta)$, and moreover, $type(\alpha \cdot \beta) = type(\alpha \cdot \gamma)$;
- if φ is of the backward form, then $\alpha \in Paths(\Delta)$, $\alpha \cdot \beta \cdot \gamma \in Paths(\Delta)$, and moreover, $type(\alpha) = type(\alpha \cdot \beta \cdot \gamma)$.

The path α is called the *prefix* of φ. The paths α, β and γ are denoted by $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$, respectively.

We denote the set of all path constraints over Δ by $P_c(\Delta)$. ∎

Definition 7.10: A *word constraint* φ *over schema* Δ is a sentence of the form

$$\forall x\ (\alpha(r, x) \rightarrow \beta(r, x)),$$

where α and β are in $Paths(\Delta)$, and $type(\alpha) = type(\beta)$. We denote α, β as $lt(\varphi)$ and $rt(\varphi)$, respectively.

We use $P_w(\Delta)$ to denote the set of all word constraints over Δ. ∎

When Δ is understood from the context, we write $P_c(\Delta)$ and $P_w(\Delta)$ simply as P_c and P_w.

We borrow the standard definition of models from first-order logic [39]. Let G be a structure in $\mathcal{U}(\Delta)$ and φ a constraint in $P_c(\Delta)$. Then we write $G \models \varphi$ if G is a model of φ.

Example 7.4: The sentences

$$\phi = \forall x\,(*(r,\ x) \rightarrow * \cdot spouse(r,\ x))$$
$$\varphi = \forall x\,(* \cdot spouse(r,\ x) \rightarrow *(r,\ x))$$
$$\psi = \forall x\,(*(r,\ x) \rightarrow \forall y\,(spouse(x,\ y) \rightarrow spouse(y,\ x)))$$

are path constraints over the schema Δ_0 given in Example 7.1. In particular, ϕ and φ are word constraints over Δ_0. Let G be the structure given in Figure 7.1. It is easy to verify that $G \models \phi \wedge \varphi \wedge \psi$.

In an instance $(\pi,\ \mu,\ d)$ of the schema, these constraints are interpreted as

$$\forall x\,(x \in d \rightarrow \exists y\,(y \in d \wedge y.spouse = x)),$$

$$\forall\, x\,(\exists\, y\,(y \in d \wedge y.spouse = x) \rightarrow x \in d),$$
$$\forall\, x\,(x \in d \rightarrow \forall\, y\,(x.spouse = y \rightarrow x = y.spouse)),$$

respectively. Here $y.spouse$ stands for the projection of record y at attribute *spouse*, and d is a subset of $\pi(Person)$. The constraint ϕ states: "each person in the set d is the spouse of someone in d", and φ states: "if a person is the spouse of someone in d, then the person is in d". The constraint ψ specifies an inverse relation: "for each person x in d, if x is the spouse of y, then y is the spouse of x". ∎

As illustrated by the example above, path constraints over a schema Δ can be naturally interpreted in database instances of Δ. Likewise, the notion "$I \models \varphi$" can also be defined for an instance I of Δ and a constraint φ of $P_c(\Delta)$.

The agreement between databases and their abstraction with respect to path constraints is revealed by the following lemma, which justifies the abstraction of structured databases defined above.

Lemma 7.1: Let Δ be a schema. For each $I \in \mathcal{I}(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$, such that

$$(\dagger) \qquad \text{for every } \varphi \in P_c(\Delta),\ I \models \varphi \text{ iff } G \models \varphi.$$

Similarly, for each $G \in \mathcal{U}_f(\Delta)$, there is $I \in \mathcal{I}(\Delta)$, such that $(\dagger)$ holds. ∎

Proof: Let $\Delta = (\mathcal{C},\ \nu,\ DBtype)$.

(1) Given $I \in \mathcal{I}(\Delta)$, we construct $G \in \mathcal{U}_f(\Delta)$, such that for each $\varphi \in P_c(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

Let $I = (\pi,\ \mu,\ d)$. Then we define V to be the smallest set satisfying the following conditions:

1. $d \in V$;
2. for every $v \in V$,
   - if v is a set (or v is an oid and $\mu(\nu)$ is a set), then every element of v (or $\mu(\nu)$) is in V;

- if v is a record (or v is an oid and $\mu(v)$ is a record), then every attribute of v (or $\mu(v)$) is in V.

For every $v \in V$, let $o(v)$ be a distinct node. Let $G = (|G|, r^G, E^G, R^G)$, where

- $|G| = \{o(v) \mid v \in V\}$;
- $r^G = o(d)$;
- for each $o(v) \in |G|$ and $\tau \in T(\Delta)$, $G \models R^G_\tau(o(v))$ iff v is of type τ;
- for all $o(v), o(v') \in |G|$,
  - $G \models *(o(v), o(v'))$ iff $v' \in v$ (or $v' \in \mu(v)$ if v is an oid),
  - for each $l \in \mathcal{L} \cap E(\Delta)$, $G \models l(o(v), o(v'))$ iff $v' = v.l$ (or $v' = \mu(v).l$ if v is an oid). Here $v.l$ means the projection of v at attribute l, i.e., the l component of v.

It is straightforward to verify the following:

- $G \in \mathcal{U}_f(\Delta)$; that is, G is a finite $\sigma(\Delta)$-structure and $G \models \Phi(\Delta)$;
- for each $\varphi \in P_c(\Delta)$, $G \models \varphi$ iff $I \models \varphi$. This can be easily verified by *reductio*.

(2) Given $G = (|G|, r^G, E^G, R^G)$ in $\mathcal{U}_f(\Delta)$, we define $I = (\pi, \mu, d)$, such that $I \in \mathcal{I}(\Delta)$ and for every $\varphi \in P_c(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

To do this, for each base type $b \in T(\Delta)$, we define an injective mapping $g_b : R^G_b \to D_b$, where R^G_b is the unary relation in G denoting the sort b, and D_b is the domain of b. By the definition of the type constraint determined by b given earlier and since G satisfies the constraint, such a mapping always exists. We substitute $g_b(o)$ for each o in R^G_b.

The instance I is defined as follows.

- for each $C \in C$, $\pi(C) = R^G_C$;
- for each $o \in \pi(C)$,

- if $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $\mu(o) = [l_1 : o_1, \ldots, l_n : o_n]$, where for each $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$;
- if $\nu(C) = \{\tau\}$, then $\mu(o) = \{o' \mid o' \in |G|,\ G \models *(o, o')\}$;

- if $DBtype = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then let $d = [l_1 : o_1, \ldots, l_n : o_n]$, where for each $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(r, o_i)$;

  if $DBtype = \{\tau\}$, then let $d = \{o' \mid o' \in |G|,\ G \models *(r, o')\}$.

Note that this is well-defined since $G \models \Phi(\Delta)$. It is easy to verify that $I \in \mathcal{I}(\Delta)$, and in addition, $G \models \varphi$ iff $I \models \varphi$.

This proves Lemma 7.1. ∎

From the lemma follows immediately the corollary below.

Corollary 7.2: Let Δ be a schema and $\Sigma \cup \{\varphi\}$ a finite subset of $P_c(\Delta)$. Then the following statements are equivalent:

1. There is $I \in \mathcal{I}(\Delta)$ such that $I \models \bigwedge \Sigma \wedge \neg\varphi$.
2. There is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$.

∎

Proof: Suppose that there is $I \in \mathcal{I}(\Delta)$ such that $I \models \bigwedge \Sigma \wedge \neg\varphi$. By Lemma 7.1, there is G in $\mathcal{U}_f(\Delta)$, such that for each $\psi \in \Sigma \cup \{\varphi\}$, $I \models \psi$ iff $G \models \psi$. Therefore, $G \models \bigwedge \Sigma \wedge \neg\varphi$.

Conversely, suppose that there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$. Again by Lemma 7.1, there is $I \in \mathcal{I}(\Delta)$, such that for each $\psi \in \Sigma \cup \{\varphi\}$, $G \models \psi$ iff $I \models \psi$. Therefore, $I \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

Because of Corollary 7.2, we can describe path constraint implication in terms of logic structures instead of database instances.

7.1.4 Path constraint implication revisited

In the context of structured databases, path constraint implication is restricted by a schema. More specifically, let Δ be a schema in $\mathcal{M}_f^+$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_c(\Delta)$. Then We use $\Sigma \models_\Delta \varphi$ to denote that Σ *implies* φ *over* Δ. That is, for every $G \in \mathcal{U}(\Delta)$, if $G \models \Sigma$ then $G \models \varphi$. Similarly, we use $\Sigma \models_{(f,\Delta)} \varphi$ to denote that Σ *finitely implies* φ *over* Δ. That is, for every $G \in \mathcal{U}_f(\Delta)$, if $G \models \Sigma$ then $G \models \varphi$. We write $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\Delta)} \varphi$) as $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) when Δ is clear from the context.

The *implication problem for* $P_c(\Delta)$ *over schema* Δ is the problem of determining, given any finite set $\Sigma \cup \{\varphi\} \subseteq P_c(\Delta)$, whether $\Sigma \models_\Delta \varphi$. Similarly, the *finite implication problem for* $P_c(\Delta)$ *over schema* Δ is the problem of determining, given any finite set $\Sigma \cup \{\varphi\} \subseteq P_c(\Delta)$, whether $\Sigma \models_{(f,\Delta)} \varphi$.

The *(finite) implication problem for path constraints* (P_c) *in the context of* $\mathcal{M}_f^+$ is the problem to determine, given any schema Δ in $\mathcal{M}_f^+$, whether the (finite) implication problem for $P_c(\Delta)$ over schema Δ is decidable.

Similarly, the (finite) implication problem for $P_w(\Delta)$ over schema Δ and the (finite) implication problem for word constraints (P_w) in the context of $\mathcal{M}_f^+$ can be defined.

7.2 The model $\mathcal{M}^+$

To further explore the impact of type systems on path constraint implication, this section introduces another object-oriented model, $\mathcal{M}^+$.

The model $\mathcal{M}^+$ is a mild variant of $\mathcal{M}_f^+$. The difference between these two is that $\mathcal{M}^+$ supports the set construct, whereas $\mathcal{M}_f^+$ supports the finite set construct. As a result, $\mathcal{M}^+$ allows infinite sets, while sets in $\mathcal{M}_f^+$ must be finite.

Syntactically, types, schemas and instances in $\mathcal{M}^+$ are defined in the same way as in $\mathcal{M}_f^+$. However, the semantic interpretation of set types in $\mathcal{M}^+$ is different from that in $\mathcal{M}_f^+$.

More specifically, in $\mathcal{M}^+$, the domain of a set type $\{\tau\}$ is defined to be

$$[\![\{\tau\}]\!]_\pi = \{V \mid V \subseteq [\![\tau]\!]_\pi\}.$$

As a result, infinite sets are allowed in $\mathcal{M}^+$.

Given a schema Δ in $\mathcal{M}^+$, the notions of $E(\Delta)$, $T(\Delta)$, $R(\Delta)$, $\sigma(\Delta)$, and type constraint $\Phi(\Delta)$ can be defined in the same way as in $\mathcal{M}_f^+$. Along the same lines, the notions of abstract databases of Δ and $\mathcal{U}_f(\Delta)$ can also be defined. However, the definition of $\mathcal{U}(\Delta)$ is different. Here $\mathcal{U}(\Delta)$ denotes the set of all the $\sigma(\Delta)$-structures satisfying $\Phi(\Delta)$. The structures in $\mathcal{U}(\Delta)$ are not required to respect the finite set rule. That is, nodes representing sets in such structures are allowed to have infinitely many outgoing edges.

As in $\mathcal{M}_f^+$, given a schema Δ in $\mathcal{M}^+$, the notions of $Paths(\Delta)$, $P_c(\Delta)$, $P_w(\Delta)$, $\models_\Delta$, $\models_{(f,\Delta)}$, and the implication and finite implication problems for $P_c(\Delta)$ and $P_w(\Delta)$ over Δ can be defined. Similarly, the implication and finite implication problems for path constraints (P_c) and word constraints (P_w) in the context of $\mathcal{M}^+$ can also be defined.

It can be shown that Lemma 7.1 also holds in $\mathcal{M}^+$.

The following should be noted.

- For any schema Δ in $\mathcal{M}^+$, $\mathcal{U}(\Delta)$ is definable in first-order logic. As a result, if the implication problem and the finite implication problem for path constraints coincide in $\mathcal{M}^+$, then both problems are decidable. However, in the next chapter, we shall show that in $\mathcal{M}^+$, these problems are different.

- In contrast, for a schema Δ in $\mathcal{M}_f^+$, if $T(\Delta)$ contains set types, then $\mathcal{U}(\Delta)$ is not definable in first-order logic. As a result, the equivalence of the implication problem and the finite implication problem for path constraints in $\mathcal{M}_f^+$ does not necessarily lead to the decidability of these problems. In the next chapter, we shall show that in $\mathcal{M}_f^+$, it is indeed the case where the implication problem and the finite implication problem for path constraints are equivalent, but these problems are undecidable.

7.3 The model $\mathcal{M}$

Next, we introduce a restriction of $\mathcal{M}_f^+$, $\mathcal{M}$. In the next chapter, we shall show that in the context of $\mathcal{M}$, path constraint implication has low complexity.

The model $\mathcal{M}$ supports classes, the record construct and recursive structures. However, it does not allows sets. More specifically, the type system of $\mathcal{M}$ is defined as follows.

Definition 7.11: Let $\mathcal{C}$ be some finite set of *classes*. The set of *types over* $\mathcal{C}$, $Types^{\mathcal{C}}$, is defined by the syntax:

$$\begin{aligned} t &::= b \mid C \\ \tau &::= t \mid [l_1 : t_1, \ldots, l_n : t_n] \end{aligned}$$

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$. ∎

The notions of schemas and instances in $\mathcal{M}$ can be defined in the same way as in Section 7.1.

Example 7.5: An example schema in $\mathcal{M}$ is $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ consists of a single class $person$,
- ν maps $person$ to a record type:

  $[Name : string, Father : person, Mother : person, Spouse : person, Likes : person]$

- $DBtype$ is $[John : person, Mary : person]$.

An instance of the schema is (π, μ, d), where

- $\pi(person)$ is a set, including $p_1, p_2, p_3, p_4, p_5, p_6, \ldots$
- the function

  $$\mu : \pi(person) \rightarrow [\![[Name : string, Father : person, Mother : person, Spouse : person, Likes : person]]\!]_\pi$$

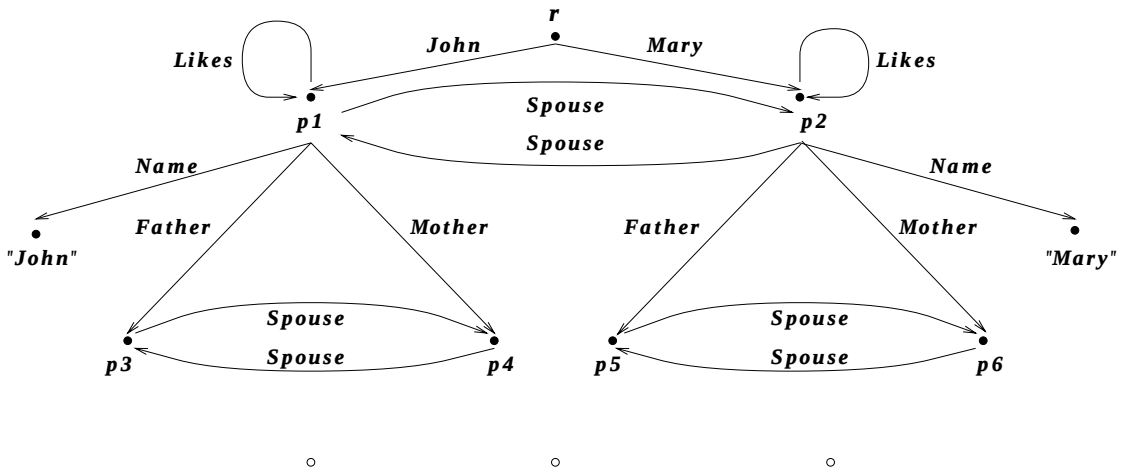

Figure 7.3: An example of abstract databases in $\mathcal{M}$

is given by:

$$p_1 \mapsto [Name : \text{“John”}, Father : p_3,\ Mother : p_4,\ Spouse : p_2,\ Likes : p_1]$$

$$p_2 \mapsto [Name : \text{“Mary”}, Father : p_5,\ Mother : p_6,\ Spouse : p_1,\ Likes : p_2]$$

$\ldots$

- $d = [John : p_1,\ Mary : p_2]$. This database can only be accessed via d.

A database of this schema represents the family trees of John and Mary. ■

Databases of $\mathcal{M}$ are comparable to feature structures studied in feature logic [72]. Feature structures have proven useful for representing linguistic data.

Given a schema Δ in $\mathcal{M}$, the notions of $E(\Delta)$, $T(\Delta)$, $R(\Delta)$, $\sigma(\Delta)$, and type constraint $\Phi(\Delta)$ can be defined in the same way as in Section 7.1, except that set types are not considered here. Similarly, the notions of abstract databases of Δ, $\mathcal{U}_f(\Delta)$ and $\mathcal{U}(\Delta)$ can also be defined.

For example, the structure depicted in Figure 7.3 can be viewed as an abstraction of (a fragment of) the database instance given in Example 7.5.

It should be noted that for any Δ in $\mathcal{M}$, $\mathcal{U}(\Delta)$ is definable in first-order logic.

As in $\mathcal{M}_f^+$, given a schema Δ in $\mathcal{M}$, we can define $Paths(\Delta)$, $P_c(\Delta)$, $P_w(\Delta)$, $\models_\Delta$, $\models_{(f,\Delta)}$,

and the implication and finite implication problems for $P_c(\Delta)$ and $P_w(\Delta)$ over Δ. Similarly, we can also define the implication and finite implication problems for path constraints (P_c) and word constraints (P_w) in the context of $\mathcal{M}$.

It is easy to verify that Lemma 7.1 also holds in the context of $\mathcal{M}$.

Chapter 8

Path Constraint Implication in Structured Data

This chapter investigates path constraint implication in the context of the object-oriented data models introduced in the last chapter.

- In the context of $\mathcal{M}_f^+$. Section 8.1 shows that the implication and finite implication problems for path constraints (P_c) are undecidable in $\mathcal{M}_f^+$. However, the implication and finite implication problems for word constraints (P_w) are decidable. In addition, in a special case, word constraint implication is decidable in PTIME.
- In the context of $\mathcal{M}^+$. Section 8.2 shows that the undecidability and decidability results established in Section 8.1 also hold in $\mathcal{M}^+$. Some of these results, however, require different proofs.
- In the context of $\mathcal{M}$. In contrast to the undecidability results established in the previous sections, Section 8.3 shows that in $\mathcal{M}$, the implication and finite implication problems for P_c are not only decidable in cubic-time, but are also finitely axiomatizable.

8.1 In the context of $\mathcal{M}_f^+$

The main results of this section are the following.

Theorem 8.1: In the context of $\mathcal{M}_f^+$, the implication and finite implication problems for path constraints (P_c) are undecidable. ∎

Theorem 8.2: In the context of $\mathcal{M}_f^+$, the implication and finite implication problems for word constraints (P_w) are decidable. ∎

We first show Theorem 8.1, and then investigates word constraint implication. In particular, we show that in a special case, word constraint implication is decidable in PTIME.

8.1.1 The undecidability of the implication problems for P_c

We show the undecidability of the finite implication problem for P_c by reduction from the word problem for finite monoids. To establish the undecidability of the implication problem for P_c, we show that in $\mathcal{M}_f^+$, the implication problem and the finite implication problem for P_c coincide. As mentioned in Section 7.2, the equivalence of the implication problem and the finite implication problem for P_c in $\mathcal{M}_f^+$ does not lead to the decidability of these problems. This is because these problems are considered in connection with some schema Δ in $\mathcal{M}_f^+$, and $\mathcal{U}(\Delta)$ is not definable in first-order logic.

We first show the undecidability of the finite implication problem for P_c. To do this, we present an encoding of the word problem for finite monoids in terms of the finite implication problem for P_c. We then show that the encoding is indeed a reduction.

Recall the description of the word problems for finite monoids given in Section 6.4. Let $, _0$ be a finite alphabet and Θ_0 be a finite set of equations (over $, _0$). Without loss of generality, assume

$$\begin{aligned} , _0 &= \{l_j \mid j \in [1,m],\ l_i \neq l_j \text{ if } i \neq j\}, \\ \Theta_0 &= \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in , _0^*,\ i \in [1,n]\}. \end{aligned}$$

Using $, _0$, we define a schema Δ_0 in $\mathcal{M}_f^+$ as follows:

$$\Delta_0 = (\mathcal{C},\ \nu,\ DBtype),$$

where

- $\mathcal{C} = \{C, C_s\}$,
- ν is defined by:

$$\begin{aligned} C &\mapsto [l_1 : C,\ \ldots,\ l_m : C] \\ C_s &\mapsto \{C\} \end{aligned}$$

- $DBtype = [a : C,\ b : C_s]$, where $a, b \notin \ ,_0$.

Note here that each symbol in $,_0$ is a record label of C, and thus is in $E(\Delta_0)$, which is the set of binary relation symbols determined by Δ_0. Moreover, every α in $,_0^*$ can be represented as a path over Δ_0, also denoted by α. In addition, it is straightforward to verify the following lemma using the type constraint $\Phi(\Delta_0)$ determined by Δ_0.

Lemma 8.3: For each $G \in \mathcal{U}(\Delta_0)$ and every $\alpha \in ,_0^*$, G has the following properties.

1. There is a unique $o \in |G|$ such that $G \models a \cdot \alpha(r^G, o)$. This node is denoted by o_α.
2. For any $o \in |G|$, if $G \models R_C^G(o)$, then there is a unique $o' \in |G|$ such that $G \models \alpha(o, o')$.

■

Next, we encode equations over $,_0$. We encode Θ_0 in terms of two finite subsets Σ_1 and Σ_2 of $P_c(\Delta_0)$.

1. The constraint

$$\forall\, x\, (a(r, x) \rightarrow b \cdot *(r, x))$$

   is in Σ_1. Moreover, for each $j \in [1, m]$, the following constraint is in Σ_1:

$$\forall\, x\, (b \cdot * \cdot l_j(r, x) \rightarrow b \cdot *(r, x))$$

   Σ_1 consists of only the constraints defined above.

2. For each $(\alpha_i,\ \beta_i) \in \Theta_0$, the following constraint is in Σ_2:

$$\forall\, x\, (b \cdot *(r, x) \rightarrow \forall\, y\, (\alpha_i(x, y) \rightarrow \beta_i(x, y)))$$

   Σ_2 consists of only the constraints defined above.

We encode a test equation $(\alpha,\ \beta)$ over $,_0$ as a constraint in $P_c(\Delta_0)$:

$$\varphi_{(\alpha,\beta)} = \forall\, x(a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x)).$$

We reduce the word problem for finite monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models_{(f,\, \Delta_0)} \varphi_{(\alpha,\beta)}.$$

It should be noted that all the constraints in the encoding are forward constraints of $P_c(\Delta_0)$.

Before we show that this is indeed a reduction, we first present some basic properties of $\mathcal{U}(\Delta_0)$ and Σ_1, which can be easily verified by using Lemma 8.3.

Lemma 8.4: For every $G \in \mathcal{U}(\Delta_0)$ and all $\alpha, \beta \in , \overset{*}{0}$, if

$$G \models \forall x\,(b \cdot *(r,x) \rightarrow \forall y\,(\alpha(x,y) \rightarrow \beta(x,y))),$$

then

$$G \models \forall x\,(b \cdot *(r,x) \rightarrow \forall y\,(\beta(x,y) \rightarrow \alpha(x,y))). \qquad \text{(a)}$$

Similarly, if

$$G \models \forall x\,(a \cdot \alpha(r,x) \rightarrow a \cdot \beta(r,x)),$$

then

$$G \models \forall x\,(a \cdot \beta(r,x) \rightarrow a \cdot \alpha(r,x)). \qquad \text{(b)}$$

■

Proof: We show (a) only. The proof of (b) is similar.

By Lemma 8.3, for each $o \in |G|$ such that $G \models b \cdot *(r^G, o)$, there is a unique node $o_1 \in |G|$ such that $G \models \alpha(o, o_1)$. Similarly, there is a unique node $o_2 \in |G|$ such that $G \models \beta(o, o_2)$. This is because $o \in R_C^G$. By $G \models \forall x\,(b \cdot *(r,x) \rightarrow \forall y\,(\alpha(x,y) \rightarrow \beta(x,y)))$, we have

$$o_1 = o_2.$$

Hence $G \models \forall x\,(b \cdot *(r,x) \rightarrow \forall y\,(\beta(x,y) \rightarrow \alpha(x,y)))$. ■

Lemma 8.5: For every $G \in \mathcal{U}(\Delta_0)$, if $G \models \Sigma_1$, then for every $\alpha \in , \overset{*}{0}$,

$$G \models b \cdot *(r^G, o_\alpha),$$

where o_α is the unique node in $|G|$ such that $G \models a \cdot \alpha(r^G, o_\alpha)$. In addition, for all $o, o' \in |G|$ such that $G \models b \cdot *(r^G,\ o') \wedge \alpha(o', o)$, we have

$$G \models b \cdot *(r^G, o).$$

■

Proof: By a straightforward induction on $|\alpha|$. ■

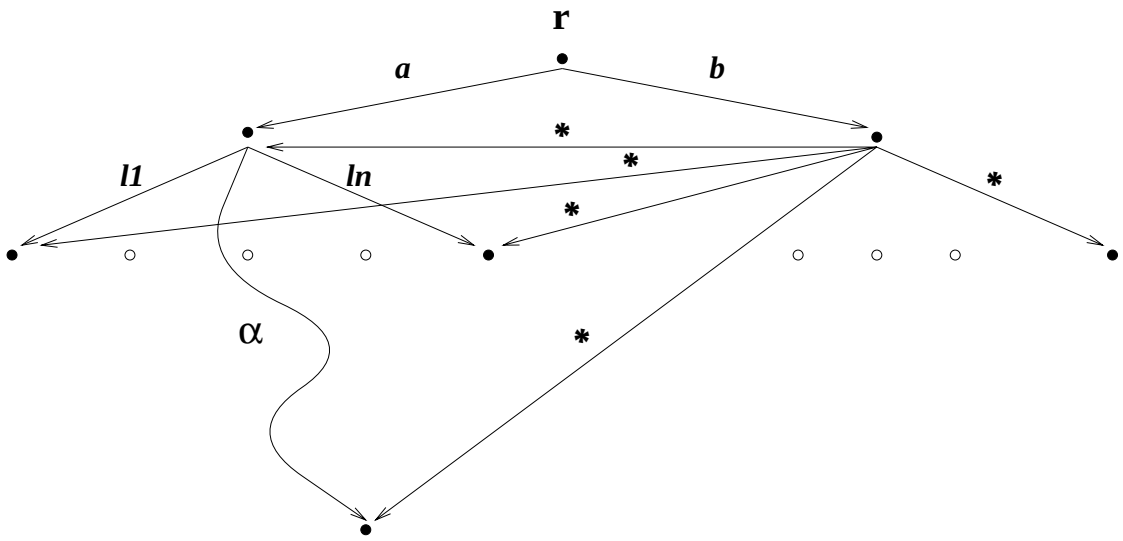

Figure 8.1: A structure in $\mathcal{U}(\Delta_0)$ that satisfies Σ_1

By Lemma 8.5, the structures in $\mathcal{U}(\Delta_0)$ that satisfy Σ_1 have the form shown in Figure 8.1.

Lemma 8.6: For every $G \in \mathcal{U}(\Delta_0)$, if $G \models \Sigma_1$ and

$$G \models \forall x\,(b \cdot *(r, x) \rightarrow \forall y\,(\alpha(x, y) \rightarrow \beta(x, y)))$$

for some $\alpha, \beta \in \, , \overset{*}{_0}$, then

$$G \models \forall x\,(a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x)).$$

■

Proof: By Lemma 8.3, there is a unique node $o \in |G|$ such that

$$G \models a(r^G,\, o).$$

Denote this node by o_a. Again by Lemma 8.3, there is a unique node $o_\alpha \in |G|$ such that

$$G \models a \cdot \alpha(r^G, o_\alpha),$$

and there is a unique node $o_\beta \in |G|$ such that

$$G \models a \cdot \beta(r^G, o_\beta).$$

Therefore, $G \models \alpha(o_a,\, o_\alpha) \wedge \beta(o_a,\, o_\beta)$. By Lemma 8.5,

$$G \models b \cdot *(r^G,\, o_a).$$

As a result, if $G \models \forall\, x\, (b \cdot *(r, x) \rightarrow \forall\, y\, (\alpha(x, y) \rightarrow \beta(x, y)))$, then

$$o_\alpha = o_\beta.$$

Hence $G \models \forall\, x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x))$. ∎

Finally, we show that the encoding given above is indeed a reduction from the word problem for finite monoids. It suffices to show the following lemma.

Lemma 8.7: In the context of $\mathcal{M}_f^+$, for all α and β in $,\, {}_0^*$,

$$\Theta_0 \models_f (\alpha,\, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_{(f,\, \Delta_0)} \forall\, x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x)).$$

∎

Proof:

(*if*) Suppose that $\Theta_0 \not\models_f (\alpha,\, \beta)$. Then we show that there exists structure $G \in \mathcal{U}_f(\Delta_0)$, such that $G \models \Sigma_1 \cup \Sigma_2$, but $G \not\models \forall\, x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x))$.

By $\Theta_0 \not\models_f (\alpha,\, \beta)$, there exist a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : ,\, {}_0^* \rightarrow M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Using M and h, we define an equivalence relation $\approx$ on $,\, {}_0^*$ as follows:

$$\rho \approx \varrho \quad \text{iff} \quad h(\rho) = h(\varrho).$$

For every $\rho \in ,\, {}_0^*$, let $\widehat{\rho}$ be the equivalence class of ρ with respect to $\approx$. Let

$$C_{\Theta_0} = \{\widehat{\rho} \mid \rho \in ,\, {}_0^*\}.$$

Using C_{Θ_0}, we construct structure $G = (|G|,\, r^G,\, E^G,\, R^G)$ as follows.

(1) $|G|$.

For each $\widehat{\rho} \in C_{\Theta_0}$, let $o(\widehat{\rho})$ be a distinct node. In addition, let o_r and o_b be two distinct nodes. Then we define

$$|G| = \{o_r, o_b\} \cup \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}.$$

(2) $r^G = o_r$.

(3) The unary relations are populated as follows.

- $R^G_C = \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}$.
- $R^G_{C_s} = \{o_b\}$.
- $R^G_{DBtype} = \{o_r\}$.

(4) The binary relations are populated as follows.

- $G \models a(o_r,\, o(\widehat{\epsilon}))$.
- $G \models b(o_r,\, o_b)$.
- For each $\widehat{\rho} \in C_{\Theta_0}$, let $G \models *(o_b,\, o(\widehat{\rho}))$.

  In addition, for each $j \in [1, m]$, let $G \models l_j(o(\widehat{\rho}),\, o(\widehat{\rho \cdot l_j}))$.

By the construction of G, it is easy to verify the following claims.

Claim 1: $G \in \mathcal{U}_f(\Delta_0)$.

To see this, first note that if $\rho \approx \varrho$, then $\rho \cdot l_j \approx \varrho \cdot l_j$ for all ρ and ϱ in $\overset{*}{,}_0$ and $j \in [1, m]$. This is because h is a homomorphism, and as a result, if $h(\rho) = h(\varrho)$, then

$$\begin{aligned} h(\rho \cdot l_j) &= h(\rho) \circ h(l_j) \\ &= h(\varrho) \circ h(l_j) \\ &= h(\varrho \cdot l_j). \end{aligned}$$

Second, C_{Θ_0} is finite. To see this, consider a function $f : C_{\Theta_0} \to M$ defined by

$$f :\ \widehat{\rho} \mapsto h(\rho).$$

Clearly, f is well-defined, total and injective. Therefore, because M is finite, C_{Θ_0} is also finite. Therefore, G is finite.

Claim 2: $G \models \Sigma_1$.

This is immediate from the construction of G.

Claim 3: $G \models \Sigma_2$.

First, by assumption, for every $i \in [1, n]$, $\alpha_i \approx \beta_i$. In addition, for every $\rho \in {}^*_0$,

$$h(\rho \cdot \alpha_i) = h(\rho \cdot \beta_i).$$

This is because h is a homomorphism. Therefore, $\rho \cdot \alpha_i \approx \rho \cdot \beta_i$. That is,

$$\widehat{\rho \cdot \alpha}_i = \widehat{\rho \cdot \beta}_i.$$

Second, by the construction of G, for any $o \in |G|$, if $G \models b \cdot *(o_r, o)$, then $o = o(\widehat{\rho})$ for some $\rho \in {}^*_0$. Moreover, by Lemma 8.3, for each $\varrho \in {}^*_0$, there is a unique $o' \in |G|$, such that

$$G \models \varrho(o(\widehat{\rho}), o').$$

By a straightforward induction on $|\varrho|$, it can be shown that

$$o' = o(\widehat{\rho \cdot \varrho}).$$

Therefore, for each $o(\widehat{\rho})$ such that $G \models b \cdot *(o_r, o(\widehat{\rho}))$, $o(\widehat{\rho \cdot \alpha}_i)$ is the unique node in $|G|$ such that

$$G \models \alpha_i(o(\widehat{\rho}), o(\widehat{\rho \cdot \alpha}_i)).$$

Similarly, we have $G \models \beta_i(o(\widehat{\rho}), o(\widehat{\rho \cdot \beta}_i))$. By $o(\widehat{\rho \cdot \alpha}_i) = o(\widehat{\rho \cdot \beta}_i)$, we have

$$G \models \beta_i(o(\widehat{\rho}), o(\widehat{\rho \cdot \alpha}_i)).$$

Therefore, for each $i \in [1, n]$,

$$G \models \forall x\, (b \cdot *(r, x) \rightarrow \forall y\, (\alpha_i(x, y) \rightarrow \beta_i(x, y))).$$

That is, $G \models \Sigma_2$.

Claim 4: $G \not\models \forall x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x))$.

As in Claim 3, we can show that

$$\begin{aligned} G &\models a \cdot \alpha(o_r, o(\widehat{\alpha})), \\ G &\models a \cdot \beta(o_r, o(\widehat{\beta})). \end{aligned}$$

In addition, $o(\widehat{\beta})$ is the unique node in $|G|$ such that $G \models a \cdot \beta(o_r,\, o(\widehat{\beta}))$. By assumption, we have $\alpha \not\approx \beta$. Thus $\widehat{\alpha} \neq \widehat{\beta}$. Hence $o(\widehat{\alpha}) \neq o(\widehat{\beta})$. Therefore,

$$G \models a \cdot \alpha(o_r,\, o(\widehat{\alpha})) \wedge \neg a \cdot \beta(o_r,\, o(\widehat{\alpha})).$$

That is, $G \not\models \forall\, x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x))$.

(*only if*) Suppose that there exists $G \in \mathcal{U}_f(\Delta_0)$, such that $G \models \Sigma_1 \cup \Sigma_2$, but

$$G \not\models \forall\, x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x)).$$

Then we show that $\Theta_0 \not\models_f (\alpha,\, \beta)$. That is, we show that there exist a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : {}^{*}_{0} \rightarrow M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$. However, $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation on ${}^{*}_{0}$ as follows:

$$\rho \sim \varrho \quad \text{iff} \quad G \models \forall\, x(b \cdot *(r, x) \rightarrow \forall\, y\, (\rho(x, y) \rightarrow \varrho(x, y))).$$

Note that by Lemma 8.4, it can be easily verified that $\sim$ is indeed an equivalence relation.

By $G \models \Sigma_2$ and Lemma 8.4 (a), for every $i \in [1, n]$,

$$\alpha_i \sim \beta_i.$$

In addition, because $G \not\models \forall\, x\, (a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x))$, by Lemma 8.4 (b) and Lemma 8.6, we have $G \not\models \forall\, x\, (b \cdot *(r, x) \rightarrow \forall\, y\, (\alpha(x, y) \rightarrow \beta(x, y)))$. Therefore,

$$\alpha \not\sim \beta.$$

For every $\rho \in {}^{*}_{0}$, let $[\rho]$ be the equivalence class of ρ with respect to $\sim$. Then clearly, for every $i \in [1, n]$, $[\alpha_i] = [\beta_i]$, but $[\alpha] \neq [\beta]$.

Using the notion of $\sim$, we define

$$M = \{[\rho] \mid \rho \in {}^{*}_{0}\}.$$

It can be shown that M has the following property.

Claim 5: M is finite.

To show this, for every $\rho \in {}^{*}_{0}$, let

$$S_\rho = \{(a, b) \mid a, b \in |G|,\ G \models b \cdot *(r^G,\ a),\ G \models \rho(a,\ b)\}.$$

In addition, let

$$S_G = \{S_\rho \mid \rho \in {}^{*}_{0}\}.$$

By Lemma 8.3, S_ρ is a finite function from $|G|$ to $|G|$. Since $|G|$ is finite, there are finitely many such functions. Therefore, S_G is finite. Moreover, it is easy to verify that for all $\rho, \varrho \in {}^{*}_{0}$,

$$\rho \sim \varrho \quad \text{iff} \quad S_\rho = S_\varrho.$$

Consider a function $g : M \to S_G$ defined by

$$g :\ [\rho] \mapsto S_\rho.$$

Clearly, g is well-defined, total and injective. Therefore, because S_G is finite, M is also finite.

Next, we define a binary operation $\circ$ on M by

$$[\rho] \circ [\varrho] = [\rho \cdot \varrho].$$

It is easy to verify the following claims.

Claim 6: $\circ$ is well-defined.

To see this, for all $\rho_1, \rho_2, \varrho_1, \varrho_2 \in {}^{*}_{0}$ such that $\rho_1 \sim \rho_2$ and $\varrho_1 \sim \varrho_2$, we show that

$$\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2.$$

By lemma 8.3, for every $o \in |G|$ such that $G \models b \cdot *(r^G,\ o)$, there exists a unique $o_1 \in |G|$ such that

$$G \models \rho_1 \cdot \varrho_1(o,\ o_1).$$

In addition, there is a unique $o' \in |G|$ such that

$$G \models \rho_1(o,\ o') \wedge \varrho_1(o',\ o_1).$$

By $\rho_1 \sim \rho_2$, we have $G \models \rho_2(o,\, o')$. By Lemma 8.5 and $G \models b \cdot *(r^G,\, o) \wedge \rho_1(o,\, o')$, we have $G \models b \cdot *(r^G,\, o')$. Thus by $\varrho_1 \sim \varrho_2$, we also have $G \models \varrho_2(o',\, o_1)$. Hence

$$G \models \rho_2 \cdot \varrho_2(o,\, o_1).$$

Therefore,

$$G \models \forall\, x\, (b \cdot *(r, x) \rightarrow \forall\, y\, (\rho_1 \cdot \varrho_1(x,\, y) \rightarrow \rho_2 \cdot \varrho_2(x, y))).$$

That is, $\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2$.

Claim 7: $\circ$ is associative.

This is because for all $[\rho], [\varrho], [\lambda] \in M$,

$$\begin{aligned} ([\rho] \circ [\varrho]) \circ [\lambda] &= [\rho \cdot \varrho] \circ [\lambda] \\ &= [\rho \cdot \varrho \cdot \lambda] \\ &= [\rho] \circ ([\varrho \cdot \lambda]) \\ &= [\rho] \circ ([\varrho] \circ [\lambda]). \end{aligned}$$

Claim 8: $[\epsilon]$ is the identity for $\circ$.

This is because for any $[\rho] \in M$,

$$[\epsilon] \circ [\rho] = [\rho] = [\rho] \circ [\epsilon].$$

By these claims, $(M,\, \circ,\, [\epsilon])$ is a finite monoid. Finally, we define $h : , \overset{*}{0} \rightarrow M$ by

$$h :\ \rho \mapsto [\rho].$$

Clearly, h is a homomorphism since

$$h(\rho \cdot \varrho) = [\rho \cdot \varrho] = [\rho] \circ [\varrho] = h(\rho) \circ h(\varrho).$$

In addition, for every $i \in [1, n]$, by $[\alpha_i] = [\beta_i]$, $h(\alpha_i) = h(\beta_i)$. Moreover, by $[\alpha] \neq [\beta]$, $h(\alpha) \neq h(\beta)$. Therefore, $\Theta_0 \not\models_f (\alpha,\, \beta)$.

This completes the proof of Lemma 8.7. ∎

From Lemma 8.7 and Theorem 6.20, the undecidability of the finite implication problem for P_c in $\mathcal{M}_f^+$ follows immediately.

Next, we show the undecidability of the implication problem for P_c. To do this, it suffices to show he following lemma.

Lemma 8.8: Let Δ be any schema in $\mathcal{M}_f^+$. For every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model in $\mathcal{U}_f(\Delta)$. ∎

For if the lemma holds, then for every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$,

$$\Sigma \models_\Delta \varphi \quad \text{iff} \quad \Sigma \models_{(f,\,\Delta)} \varphi.$$

To see this, first note that if $\Sigma \models_\Delta \varphi$, then $\Sigma \models_{(f,\,\Delta)} \varphi$. Conversely, if $\Sigma \not\models_\Delta \varphi$, then by Lemma 8.8, $\Sigma \not\models_{(f,\,\Delta)} \varphi$. Therefore, the implication problem for P_c is undecidable if and only if the finite implication problem for P_c is undecidable.

Below we show Lemma 8.8.

Proof of Lemma 8.8: Given $\Sigma \cup \{\varphi\} \subset P_c(\Delta)$ and a model G of $\bigwedge \Sigma \wedge \neg\varphi$ in $\mathcal{U}(\Delta)$, we construct a finite structure G' such that $G' \in \mathcal{U}_f(\Delta)$ and $G' \models \bigwedge \Sigma \wedge \neg\varphi$. To do this, we first define the notion of k-neighborhood of a structure, as follows (recall that a similar notion is defined in the context of semistructured data in Definition 5.5).

For each structure G in $\mathcal{U}(\Delta)$ and each natural number k, *the k-neighborhood of G* is the substructure G_k of G with its universe

$$|G_k| = \{o \mid o \in |G|,\ G \models \alpha(r^G,\, o) \text{ for some } \alpha \in Paths(\Delta) \text{ with } |\alpha| \leq k\}.$$

Given Σ and φ as described above, let

$$k = max\{|pf(\psi)| + |lt(\psi)| + |rt(\psi)| \ \mid \ \psi \in \Sigma \cup \{\varphi\}\} \ + \ 1,$$

and let G_k be the k-neighborhood of G. Here the notations pf, lt and rt are described in Definition 7.9. We construct G' as follows. For each $\tau \in T(\Delta)$, let $o(\tau)$ be a distinct node, and let $G' = (|G'|, r^{G'}, E^{G'}, R^{G'})$, where

- $|G'| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta)\}$,
- $r^{G'} = r^{G_k}$,
- for each $\tau \in T(\Delta)$, $R_\tau^{G'} = (R_\tau^G \cap |G_k|) \cup \{o(\tau)\}$,
- $E^{G'}$ is E^{G_k} augmented with the following:
  - for each $o \in R_\tau^G \cap |G_k|$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or for some class C, $\tau = C$ and $\nu(C) = [l_1 : \tau_1, ..., l_n : \tau_n]$), and if for some $i \in [1, n]$ and any $o' \in |G_k|$, $G_k \not\models l_i(o, o')$, then let $G' \models l_i(o, o(\tau_i))$;
  - for each type $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or for some class C, $\tau = C$ and $\nu(C) = [l_1 : \tau_1, ..., l_n : \tau_n]$), then for each $i \in [1, n]$, let $G \models l_i(o(\tau), o(\tau_i))$.

We now show that G' is indeed the structure desired.

(1) $G' \in \mathcal{U}_f(\Delta)$.

Since $G \in \mathcal{U}(\Delta)$, each node in $|G|$ has finitely many outgoing edges. Hence by the definition of G_k, $|G_k|$ is finite. In addition, $T(\Delta)$ is finite. Therefore, by the construction of G', $|G'|$ is finite. In addition, it can be easily verified that $G' \models \Phi(\Delta)$.

(2) $G' \models \bigwedge \Sigma \wedge \neg\varphi$.

The following can be easily verified by *reductio*:

Claim: $G \models \bigwedge \Sigma \wedge \neg\varphi$ iff $G_k \models \bigwedge \Sigma \wedge \neg\varphi$. In addition, if G_k is the k-neighborhood of G', then $G' \models \bigwedge \Sigma \wedge \neg\varphi$ iff $G_k \models \bigwedge \Sigma \wedge \neg\varphi$.

By the claim, it suffices to show that G_k is also the k-neighborhood of G'. To do this, assume for *reductio* that there exist $o(\tau) \in |G'|$ and $\alpha \in Paths(\Delta)$ such that $|\alpha| \leq k$ and $G' \models \alpha(r, o(\tau))$. Without loss of generality, assume that α has the shortest length among such paths. Then by the construction of G', there exist $o \in |G_k|$, $l \in \mathcal{L}$ and $\alpha' \in Paths(\Delta)$, such that

- $\alpha = \alpha' \cdot l$ and $G' \models \alpha'(r^{G'}, o) \wedge l(o, o(\tau))$;

- there is $\tau \in T(\Delta)$ such that $\tau = [l : \tau, ...]$ (or $\tau = C$ and $\nu(C) = [l : \tau, ...]$ for some class C), $o \in R^G_\tau$, and for any $o' \in |G_k|$, $G_k \not\models l(o, o')$; and
- $G_k \models \alpha'(r^{G'}, o)$. This is because for each $\tau \in T(\Delta)$, $o(\tau)$ does not have any outgoing edge to any node of $|G_k|$.

By $G \in \mathcal{U}(\Delta)$, there is $o' \in |G|$ such that $G \models l(o, o')$. By the argument above, $o' \notin |G_k|$. Hence by the definition of k-neighborhood, there is no path $\beta \in Paths(\Delta)$ such that $|\beta| < k$ and $G \models \beta(r, o) \wedge l(o, o')$. Therefore, α' must have a length of at least k. That is, $|\alpha| > k$. This contradicts the assumption. Hence G_k is indeed the k-neighborhood of G'.

Therefore, G' is indeed the structure desired. This proves Lemma 8.8. ∎

8.1.2 The decidability of the implication problems for P_w

Next, we study word constraint implication in the context of $\mathcal{M}^+_f$. We first prove Theorem 8.2, and then show that in a special case, word constraint implication is decidable in PTIME.

As observed by [9], word constraint implication can be expressed in two-variable first-order logic FO^2 in the context of the semistructured data model $\mathcal{SM}$. In addition, [9] showed that in $\mathcal{SM}$, both the implication and finite implication problems for word constraints are decidable in PTIME.

However, in the context of $\mathcal{M}^+_f$, word constraint implication cannot be stated in FO^2. This is because in the (finite) implication problem for $P_w(\Delta)$ over a schema Δ, each structure considered must satisfy $\Phi(\Delta)$, which is in C^2 (two-variable logic with counting) but is not in FO^2.

In addition, the proof of the decidability of word constraint implication given in [9] also breaks down in $\mathcal{M}^+_f$. The proof is established by showing that a set of inference rules, $\mathcal{I}_{AV}$, is sound and complete for word constraint implication. This set consists of Reflexivity, Transitivity and Right-congruence described in Section 6.1. However, the lemma below shows that the proof no longer holds in $\mathcal{M}^+_f$.

Lemma 8.9: In the context of $\mathcal{M}_f^+$, $\mathcal{I}_{AV}$ is not complete for word constraint implication. ∎

Proof: Consider the constraints ϕ and φ given in Example 7.4. By induction on the lengths of proofs, it can be shown that φ is not provable from ϕ using $\mathcal{I}_{AV}$. More specifically, it can be shown that if φ were provable from ϕ using $\mathcal{I}_{AV}$, then the length of $lt(\varphi)$ would be strictly less than the length of $rt(\varphi)$.

However, by the type constraint determined by the schema Δ_0 given in Example 7.1, it is indeed the case where $\{\phi\} \models_{\Delta_0} \varphi$. More specifically, consider an instance I of the schema satisfying ϕ, where $I = (\pi, \mu, d)$. Let $s = \{x.spouse \mid x \in d\}$ and let $|d|$, $|s|$ denote the cardinalities of d and s, respectively. By the type constraint determined by record type, $|s| \leq |d|$. By $I \models \phi$, $d \subseteq s$. Hence $d = s$, and therefore, $I \models \varphi$. ∎

Next, we show that in $\mathcal{M}_f^+$, word constraint implication is still decidable.

Proof of Theorem 8.2: The decidability of the finite implication follows from the decidability of the finite satisfiability problem for C^2, which was established by [15], since the type constraints are expressible in C^2 and all the word constraints are expressible in FO^2. As a result, the implication problem for word constraints is also decidable, because of Lemma 8.8. ∎

Next, we consider word constraints of the form:

$$\forall x\,(\alpha(r, x) \rightarrow \beta \cdot *(r, x)).$$

We refer to such a constraint as a constraint having *the $*$-form*. Over a schema Δ in $\mathcal{M}_f^+$, the *$*$-form (finite) implication problem for $P_w(\Delta)$* is the problem to determine, given any finite set $\Sigma \cup \{\varphi\}$ of $*$-form constraints in $P_w(\Delta)$, whether $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\,\Delta)} \varphi$).

We next show that $*$-form implication and finite implication problems for $P_w(\Delta)$ are decidable in PTIME.

Proposition 8.10: Over any schema Δ in $\mathcal{M}_f^+$, the $*$-form implication and finite impli-

cation problems for $P_w(\Delta)$ are finitely axiomatizable and are decidable in PTIME. ∎

To show this proposition, it suffices to show the following lemma.

Lemma 8.11: Over any schema Δ in $\mathcal{M}_f^+$, $\mathcal{I}_{AV}$ is sound and complete for $*$-form implication and finite implication problems for $P_w(\Delta)$. That is, for any finite set $\Sigma \cup \{\varphi\}$ of $*$-form constraints in $P_w(\Delta)$, $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$ iff $\Sigma \models_\Delta \varphi$, and $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$ iff $\Sigma \models_{(f,\Delta)} \varphi$. ∎

Here $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$ stands for that φ is provable from Σ using $\mathcal{I}_{AV}$. That is, there is an $\mathcal{I}_{AV}$-proof of φ from Σ.

For if Lemma 8.11 holds, then the PTIME decidability of $*$-form implication and finite implication follows immediately from the lemma below, which was established in [9].

Lemma 8.12 [9]: Let Σ be a finite set of word constraints and α a path. The set

$$RewriteTo(\alpha) = \{\beta \mid \Sigma \vdash_{\mathcal{I}_{AV}} \forall x\,(\alpha(r,x) \rightarrow \beta(r,x))\}$$

is a regular language recognized by a nondeterministic finite state automata constructible in polynomial time from Σ and α. In particular, whether $\Sigma \vdash_{\mathcal{I}_{AV}} \forall x\,(\alpha(r,x) \rightarrow \beta(r,x))$ can be decided in PTIME. ∎

Proof of Lemma 8.11: The soundness of $\mathcal{I}_{AV}$ can be verified by a straightforward induction on the lengths of $\mathcal{I}_{AV}$-proofs.

For the proof of the completeness, it suffices to show the following claim.

Claim 1: Given any schema Δ and finite set $\Sigma \cup \{\varphi\}$ of $*$-form constraints in $P_w(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \Sigma$, and in addition, if $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$.

For if Claim 1 holds and $\Sigma \models_\Delta \varphi$, then by $G \models \Sigma$, we have $G \models \varphi$. In addition, by $G \in \mathcal{U}_f(\Delta)$, if $\Sigma \models_{(f,\Delta)} \varphi$, then we also have $G \models \varphi$. Thus again by Claim 1, $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$.

We next show Claim 1 by constructing the structure G desired.

Let $\Delta = (\mathcal{C}, \nu, DBtype)$. We first assume that for each base type $b \in T(\Delta)$, the domain

of b is infinite. Let

$$k = max\{|lt(\psi)|, |rt(\psi)| \mid \psi \in \Sigma \cup \{\varphi\}\} + 1.$$

We define the structure G described in Claim 1 in two steps: we first define the k-neighborhood of G, G_k, and then construct G from G_k. The notion of k-neighborhood is described in the proof of Lemma 8.8.

To construct G_k, we define the following.

- $Paths^k(\Delta) = \{\alpha \mid \alpha \in Paths(\Delta),\ |\alpha| \leq k\}$.
- An equivalence relation $\approx$ on $Paths^k(\Delta)$ defined by

  $$\alpha \approx \beta \text{ iff } \Sigma \vdash_{\mathcal{I}_{AV}} \forall x\ (\alpha(r, x) \rightarrow \beta(r, x)) \text{ and } \Sigma \vdash_{\mathcal{I}_{AV}} \forall x\ (\beta(r, x) \rightarrow \alpha(r, x)).$$

- $\widehat{\alpha}$ denoting the equivalence class of α with respect to $\approx$.
- $\mathcal{A} = \{\widehat{\alpha} \mid \alpha \in Paths^k(\Delta)\}$.
- $type(\widehat{\alpha}) = type(\alpha)$, where $type(\alpha)$ is the type of path α determined by Δ. This is well-defined since if α and β are in the same equivalence class, then by the definition of $P_w(\Delta)$, $type(\alpha) = type(\beta)$.
- A partial order $\prec$ on $\mathcal{A}$ defined by

  $$\widehat{\alpha} \prec \widehat{\beta} \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_{AV}} \forall x\ (\alpha(r, x) \rightarrow \beta(r, x)).$$

  Note that this is well-defined by Transitivity in $\mathcal{I}_{AV}$.

We define $G_k = (|G_k|, r^{G_k}, E^{G_k}, R^{G_k})$ as follows.

- For each $\widehat{\alpha} \in \mathcal{A}$, let $o(\widehat{\alpha})$ be a distinct node and let $|G_k| = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A}\}$.
- Let $r^{G_k} = o(\widehat{\epsilon})$.
- For each $\tau \in T(\Delta)$, let $R_\tau^{G_k} = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A},\ type(\widehat{\alpha}) = \tau\}$.
- The binary relations in E^{G_k} are populated as follows.

- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$, then for each $i \in [1, n]$, let $G_k \models l_i(o(\widehat{\alpha}), o(\widehat{\beta \cdot l_i}))$. Note that this is well-defined by Transitivity and Right-congruence in $\mathcal{I}_{AV}$.
- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = \{\tau\}$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = \{\tau\}$), and there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$, then for each $\widehat{\gamma} \prec \widehat{\beta \cdot *}$, let $G_k \models *(o(\widehat{\alpha}), o(\widehat{\gamma}))$.

Based on G_k, we define G as follows. For each $\tau \in T(\Delta)$, let $o(\tau)$ be a distinct node. Let $G = (|G|, r^G, E^G, R^G)$, where

- $|G| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta)\}$;
- $r^G = r^{G_k}$;
- for each $\tau \in T(\Delta)$, $R^G_\tau = R^{G_k}_\tau \cup \{o(\tau)\}$;
- for each $l \in E(\Delta)$, if $G_k \models l(o, o')$, then $G \models l(o, o')$. Moreover,
  - for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and for some $i \in [1, n]$, $o(\widehat{\alpha})$ does not have any outgoing edge labeled with l_i, then let $G \models l_i(o(\widehat{\alpha}), o(\tau_i))$;
  - for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = \{\tau\}$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = \{\tau\}$), then let $G \models *(o(\widehat{\alpha}), o(\tau))$;
  - for each type $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or τ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, let $G \models l_i(o(\tau), o(\tau_i))$.

We now show that G is indeed the structure described in Claim 1.

1. $G \in \mathcal{U}_f(\Delta)$.

Obviously, $|G|$ is finite since $Paths^k(\Delta)$ and $T(\Delta)$ are finite. We next show that for each $o \in |G|$, if $o \in R^G_\tau$, then $G \models \phi_\tau(o)$. Here ϕ_τ is the constraint determined by type τ as described in Definition 7.6.

Case 1: $o = o(\tau)$.

By the construction of G, it is obvious $G \models \phi_\tau(o(\tau))$.

Case 2: $o = o(\widehat{\alpha})$.

If $type(\widehat{\alpha}) = b$ for some base type b, then by the construction of G_k, $o(\widehat{\alpha})$ does not have any outgoing edge. Thus $G \models \phi_\tau(o(\widehat{\alpha}))$.

If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), we have two cases to consider.

First, if for each $\beta \in \widehat{\alpha}$, $k \leq |\beta|$, then by the construction of G, for each $i \in [1, n]$,

$$G \models l_i(o(\widehat{\alpha}),\, o(\tau_i)),$$

and moreover, these are all the outgoing edges of $o(\widehat{\alpha})$. Clearly, $o(\tau_i) \in R^G_{\tau_i}$. Hence we have that $G \models \phi_\tau(o(\widehat{\alpha}))$.

Second, suppose that there is $\beta \in \widehat{\alpha}$, such that $|\beta| < k$. Then by the construction of G_k, for each $i \in [1, n]$,

$$G \models l_i(o(\widehat{\alpha}),\, o(\widehat{\beta \cdot l_i})).$$

By Definition 7.8, $type(\widehat{\beta \cdot l_i}) = type(\beta \cdot l_i) = \tau_i$. That is, $o(\widehat{\beta \cdot l_i}) \in R^G_{\tau_i}$. Moreover, by Right-congruence, for each $\gamma \in \widehat{\alpha}$, we have $\beta \cdot l_i \approx \gamma \cdot l_i$. Hence $o(\widehat{\alpha})$ has a unique outgoing edge labeled with l_i. Therefore, $G \models \phi_\tau(o(\widehat{\alpha}))$.

If $\tau = \{\tau'\}$ (or $\tau = C$ and $\nu(C) = \{\tau'\}$), again we consider two cases.

If for each $\beta \in \widehat{\alpha}$, $k \leq |\beta|$, then by the construction of G, $G \models *(o(\widehat{\alpha}), o(\tau'))$. In addition, $o(\widehat{\alpha})$ does not have any other outgoing edge. Clearly, $o(\tau') \in R^G_{\tau'}$. Hence $G \models \phi_\tau(o(\widehat{\alpha}))$.

Now suppose that there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$. Then by the definition of G, for each γ in $Paths^k(\Delta)$, if $\widehat{\gamma} \prec \widehat{\beta \cdot *}$, then $G \models *(o(\widehat{\alpha}),\, o(\widehat{\gamma}))$. Moreover, $G \models *(o(\widehat{\alpha}),\, o(\tau'))$. These are all the outgoing edges from $o(\widehat{\alpha})$. Therefore, $o(\widehat{\alpha})$ has finitely many outgoing edges, which are all labeled with $*$. In addition, clearly $o(\tau') \in R^G_{\tau'}$. Moreover, by $\widehat{\gamma} \prec \widehat{\beta \cdot *}$, we have $type(\widehat{\gamma}) = type(\widehat{\beta \cdot *}) = \tau'$. Hence $o(\widehat{\gamma}) \in R^G_{\tau'}$. Thus $G \models \phi_\tau(o(\widehat{\alpha}))$.

This proves that $G \models \Phi(\Delta)$, and consequently, $G \in \mathcal{U}_f(\Delta)$.

2. $G \models \Sigma$.

It suffices to show the following claim.

Claim 2: For each $\alpha \in Paths^k(\Delta)$, let

$$\begin{aligned} obj(\alpha) &= \{o \mid o \in |G_k|,\ G \models \alpha(r,\, o)\}; \\ inf(\alpha) &= \{o(\widehat{\beta}) \mid \widehat{\beta} \in \mathcal{A},\ \widehat{\beta} \prec \widehat{\alpha}\}. \end{aligned}$$

Then $obj(\alpha) = inf(\alpha)$.

To see this, assume for *reductio* that there is $\psi \in \Sigma$, where $\psi = \forall x\ (\alpha(r,\, x) \rightarrow \beta \cdot *(r,\, x))$, such that $G \not\models \psi$. That is, there is $o \in |G|$, such that $G \models \alpha(r,\, o) \wedge \neg\beta \cdot *(r,\, o)$.

If $o \in |G_k|$, then $o \in obj(\alpha)$. By $\Sigma \vdash_{\mathcal{I}_{AV}} \psi$, we have $\widehat{\alpha} \prec \widehat{\beta \cdot *}$. Hence $inf(\alpha) \subseteq \inf(\beta \cdot *)$. Therefore, by Claim 2, $obj(\alpha) \subseteq obj(\beta \cdot *)$. Hence $o \in obj(\beta \cdot *)$. That is, $G \models \beta \cdot *(r, o)$. This contradicts the assumption.

If $o \in |G| \setminus |G_k|$, i.e., $o = o(\tau)$ for some type $\tau \in T(\Delta)$, then by the definition of $P_w(\Delta)$, $type(\beta \cdot *) = type(\alpha) = \tau$. By Definition 7.8, $type(\beta) = \{\tau\}$. Since $o(\widehat{\beta}) \in inf(\beta)$, by Claim 2, $o(\widehat{\beta}) \in obj(\beta)$. That is, $G \models \beta(r,\, o(\widehat{\beta}))$. By the construction of G, $G \models *(o(\widehat{\beta}),\, o(\tau))$. Hence $G \models \beta \cdot *(r,\, o(\tau))$. This again contradicts the assumption.

Hence $G \models \Sigma$.

We next show Claim 2 by induction on $|\alpha|$.

Base case: $\alpha = \epsilon$.

Since all the constraints in Σ have the $*$-form, by $\mathcal{I}_{AV}$, it is easy to see that for each β in $Paths^k$, if $\widehat{\beta} \prec \widehat{\epsilon}$, then $\beta = \epsilon$. Therefore, $inf(\epsilon) = \{o(\widehat{\epsilon})\} = \{r^G\} = obj(\epsilon)$.

Inductive step: Assume Claim 2 for $|\alpha| < m$.

We next show the claim holds for $\alpha \cdot K$, where K is either $*$ or some record label l.

(1) $inf(\alpha \cdot K) \subseteq obj(\alpha \cdot K)$.

Let o be a node in $inf(\alpha \cdot K)$.

If $K \neq *$, then by Definition 7.8, $type(\widehat{\alpha})$ is some record type with label K. In addition, by the definition of inf, there is $\beta \in Paths^k(\Delta)$ such that $o = o(\widehat{\beta})$ and $\widehat{\beta} \prec \widehat{\alpha \cdot K}$. Since all the constraints in Σ have the $*$-form, by $\mathcal{I}_{AV}$, there must be $\beta' \in Paths^k(\Delta)$ such that

$$\beta = \beta' \cdot K \quad \text{and} \quad \widehat{\beta'} \prec \widehat{\alpha}.$$

This can be verified by a straightforward induction on the lengths of $\mathcal{I}_{AV}$-proofs of the constraint $\forall x\,(\beta(r, x) \rightarrow \alpha \cdot K(r, x))$ from Σ. Thus $o(\widehat{\beta'}) \in inf(\alpha)$. By the induction hypothesis, we have that $o(\widehat{\beta'}) \in obj(\alpha)$. That is,

$$G \models \alpha(r,\, o(\widehat{\beta'})).$$

Since $|\beta'| < |\beta| \leq k$ and $type(\beta') = type(\alpha)$, by the definition of G,

$$G \models K(o(\widehat{\beta'}), o(\widehat{\beta' \cdot K})).$$

Therefore, $o(\widehat{\beta}) \in obj(\alpha \cdot K)$. That is, $o \in obj(\alpha \cdot K)$.

If $K = *$, then by Definition 7.8, $type(\widehat{\alpha}) = \{type(\alpha \cdot *)\}$. Moreover, there is $\beta \in Paths^k(\Delta)$ such that $o = o(\widehat{\beta})$ and $\widehat{\beta} \prec \widehat{\alpha \cdot *}$. By the induction hypothesis, $o(\widehat{\alpha}) \in inf(\alpha) = obj(\alpha)$. That is,

$$G \models \alpha(r,\, o(\widehat{\alpha})).$$

Since $\widehat{\beta} \prec \widehat{\alpha \cdot *}$, by the construction of G_k,

$$G \models *(o(\widehat{\alpha}),\, o(\widehat{\beta})).$$

Hence $o(\widehat{\beta}) \in obj(\alpha \cdot *)$. That is, $o \in obj(\alpha \cdot *)$.

Therefore, $inf(\alpha \cdot K) \subseteq obj(\alpha \cdot K)$.

(2) $obj(\alpha \cdot K) \subseteq inf(\alpha \cdot K)$.

For each $o \in obj(\alpha \cdot K)$, there is $o' \in obj(\alpha)$, such that $G \models K(o',\, o)$.

If $K \neq *$, then $type(\alpha)$ is some record type with label K. By the induction hypothesis, $inf(\alpha) = obj(\alpha)$. Thus $o' \in inf(\alpha)$. Hence there is some $\beta \in Paths^k(\Delta)$, such that $\widehat{\beta} \prec \widehat{\alpha}$

and $o' = o(\widehat{\beta})$. Since $o \in |G_k|$ and $G \models K(o(\widehat{\beta}), o)$, by the construction of G_k, there must be $\gamma \in \widehat{\beta}$ such that $|\gamma| < k$ and

$$o(\widehat{\gamma \cdot K}) = o.$$

Since $\widehat{\gamma} = \widehat{\beta}$ and $\widehat{\beta} \prec \widehat{\alpha}$, by Right-congruence,

$$\widehat{\gamma \cdot K} \prec \widehat{\alpha \cdot K}.$$

Hence $o(\widehat{\gamma \cdot K}) \in inf(\alpha \cdot K)$. That is, $o \in inf(\alpha \cdot K)$.

If $K = *$, then $type(\alpha) = \{type(\alpha \cdot *)\}$. By the induction hypothesis, $inf(\alpha) = obj(\alpha)$. Thus $o' \in inf(\alpha)$. Hence there is $\beta \in Paths^k(\Delta)$ such that $\widehat{\beta} \prec \widehat{\alpha}$ and $o' = o(\widehat{\beta})$. By the definition of $P_w(\Delta)$, $type(\widehat{\beta}) = \{type(\alpha \cdot *)\}$. Since $o \in |G_k|$ and $G \models *(o(\widehat{\beta}), o)$, by the construction of G_k, there must be $\gamma \in \widehat{\beta}$ such that $|\gamma| < k$. Hence $\widehat{\gamma \cdot *} \in \mathcal{A}$. In addition, there must be $\rho \in Paths^k(\Delta)$ such that

$$\widehat{\rho} \prec \widehat{\gamma \cdot *} \text{ and } o(\widehat{\rho}) = o.$$

Since $\gamma \in \widehat{\beta}$, we have $\widehat{\gamma} = \widehat{\beta}$. Since $\widehat{\beta} \prec \widehat{\alpha}$, by Right-congruence, $\widehat{\gamma \cdot *} \prec \widehat{\alpha \cdot *}$. By Transitivity,

$$\widehat{\rho} \prec \widehat{\alpha \cdot *}.$$

Hence $o(\widehat{\rho}) \in inf(\alpha \cdot *)$. That is, $o \in inf(\alpha \cdot *)$.

Therefore, $obj(\alpha \cdot K) \subseteq inf(\alpha \cdot K)$. This proves Claim 2.

3. If $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$.

Let $\varphi = \forall x\,(\alpha(r, x) \rightarrow \beta \cdot *(r, x))$. Since α and $\beta \cdot *$ are in $Paths^k(\Delta)$ and $G \models \varphi$, we have $obj(\alpha) \subseteq obj(\beta \cdot *)$. Hence by Claim 2, we have

$$inf(\alpha) \subseteq inf(\beta \cdot *).$$

Since $o(\widehat{\alpha}) \in inf(\alpha)$, we have $o(\widehat{\alpha}) \in inf(\beta \cdot *)$. Therefore, by the definition of inf,

$$\widehat{\alpha} \prec \widehat{\beta \cdot *}$$

Hence $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$.

This shows that if the domain of each base type in $T(\Delta)$ is infinite, then Claim 1 holds.

Now suppose that some base types in $T(\Delta)$ have finite domains (assume that each of these finite domains has at least two elements). We construct a structure G' which has all the properties described in Claim 1 as follows.

Let G be the structure defined above. For each base type $b \in T(\Delta)$ with a finite domain and for all $\widehat{\alpha}$, $\widehat{\beta}$ in $\mathcal{A}$, we identify $o(\widehat{\alpha})$ with $o(\widehat{\beta})$ in $|G|$ if all the following conditions are satisfied:

- $type(\widehat{\alpha}) = type(\widehat{\beta}) = b$;
- if $\widehat{lt(\varphi)} \not\preccurlyeq \widehat{rt(\varphi)}$, then none of the following holds:
  - $\widehat{lt(\varphi)} \prec \widehat{\alpha}$ and $\widehat{\beta} \prec \widehat{rt(\varphi)}$,
  - $\widehat{lt(\varphi)} \prec \widehat{\beta}$ and $\widehat{\alpha} \prec \widehat{lt(\varphi)}$.

In addition, we equalize $o(\tau)$ with $o(\widehat{\alpha})$ for some $\widehat{\alpha} \in \mathcal{A}$ such that $\widehat{\alpha} \neq \widehat{rt(\varphi)}$. If such $\widehat{\alpha}$ does not exist, then let $o(\tau)$ be a distinct node as before.

Let G' be the structure constructed from G by equalizing nodes in $|G|$ as described above. Clearly, $|G'| \subseteq |G|$, and for each base type $b \in T(\Delta)$, if the domain of b is finite, then $R_b^{G'}$ has at most two elements. In addition, by the definition of G', it is easy to verify the following claims.

Claim 3: $G' \models \Phi(\Delta)$.

Claim 4: For each $\alpha \in Paths^k(\Delta)$ and $o \in |G'|$, if $G \models \alpha(r, o)$, then $G' \models \alpha(r, o)$.

Claim 5: If $G' \models \varphi$, then $G \models \varphi$.

These suffice for a proof of Claim 1. For by Claim 3, $G' \in \mathcal{U}_f(\Delta)$. Using Claim 4, it is easy to verify that $G' \models \Sigma$ by *reductio*. By Claim 5, if $G' \models \varphi$, then by the proof above, $\Sigma \vdash_{\mathcal{I}_{AV}} \varphi$.

This completes the proof of Lemma 8.11. ∎

8.2 In the context of $\mathcal{M}^+$

This section shows that the results established in the last section also hold in the context of $\mathcal{M}^+$.

Theorem 8.13: In the context of $\mathcal{M}^+$, the implication and finite implication problems for path constraints (P_c) are undecidable. ∎

Theorem 8.14: In the context of $\mathcal{M}^+$, the implication and finite implication problems for word constraints (P_w) are decidable. ∎

8.2.1 The undecidability of the implication problems for P_c

Recall that the only difference between $\mathcal{M}^+$ and $\mathcal{M}_f^+$ is that $\mathcal{M}_f^+$ supports finite sets, while $\mathcal{M}^+$ supports sets which are not necessarily finite. Syntactically, every schema Δ in $\mathcal{M}_f^+$ is also a schema in $\mathcal{M}^+$. In addition, $\mathcal{U}_f(\Delta)$ in $\mathcal{M}_f^+$ is the same as $\mathcal{U}_f(\Delta)$ in $\mathcal{M}^+$. However, $\mathcal{U}(\Delta)$ in $\mathcal{M}^+$ is definable in first-order logic, while $\mathcal{U}(\Delta)$ in $\mathcal{M}_f^+$ is not.

As a result, the reduction from the word problem for finite monoids given in the last section also provides a proof of the undecidability of the finite implication problem for P_c in the context of $\mathcal{M}^+$. Indeed, it is easy to verify that Lemmas 8.3, 8.4, 8.5, 8.6 and 8.7 also hold in $\mathcal{M}^+$.

In addition, in the context of $\mathcal{M}^+$, the encoding given in the last section also serves as a reduction from the word problem for monoids to the implication problem for P_c. More specifically, in $\mathcal{M}^+$, the following lemma holds:

Lemma 8.15: Let $, \overset{*}{_0}, \Theta_0, \Delta_0, \Sigma_1$ and Σ_2 be described as in the encoding given in Section 8.1. In the context of $\mathcal{M}^+$, for all α and β in $, \overset{*}{_0}$,

$$\Theta_0 \models (\alpha, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_{\Delta_0} \forall x\,(a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x)), \tag{a}$$

$$\Theta_0 \models_f (\alpha, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_{(f, \Delta_0)} \forall x\,(a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x)). \tag{b}$$

∎

The proof of (b) of the lemma is the same as the proof of Lemma 8.7. The proof of (a) is

similar and, in fact, simpler.

Theorem 8.13 follows from Lemma 8.15 immediately.

One may wonder why Lemma 8.15 (a) does not hold in the context of $\mathcal{M}_f^+$. As shown by Lemma 8.5, for any structure G in $\mathcal{U}(\Delta_0)$, if $G \models \Sigma_1$, then G has the form shown in Figure 8.1. Let o_b be the node in G such that $G \models b(r^G, o_b)$. The node o_b represents a set, and for any $o \in |G|$, if $o \notin \{r^G, o_b\}$, then

$$G \models *(o_b, o).$$

That is, all the nodes of G except r^G and o_b are elements of o_b. Therefore, if G is infinite, then o_b represents an infinite set. This is not allowed in $\mathcal{M}_f^+$. In other words, in $\mathcal{M}_f^+$, such structures cannot be in $\mathcal{U}(\Delta_0)$. However, this is not the case in the context of $\mathcal{M}^+$.

8.2.2 The decidability of the implication problems for P_w

In the context of $\mathcal{M}^+$, word constraint implication is also decidable.

Proof of Theorem 8.14: For any schema Δ in $\mathcal{M}^+$, the type constraint $\Phi(\Delta)$ is in two-variable logic with counting, C^2. In addition, the set of word constraints, $P_w(\Delta)$, is definable in two-variable first-order logic FO^2, which is a fragment of C^2. Therefore, The decidability of the finite implication problem for $P_w(\Delta)$ follows from the decidability of the finite satisfiability problem for C^2 [15]. Moreover, the decidability of the implication problem for $P_w(\Delta)$ follows from the decidability of the satisfiability problem for C^2, which was established in [46]. ∎

Again, the proof of the decidability of the implication problem for P_w above is not applicable in the context of $\mathcal{M}_f^+$. This is because over a schema Δ in $\mathcal{M}_f^+$, $\mathcal{U}(\Delta)$ may not be definable in first-order logic, and therefore, may not be definable in C^2.

In the context of $\mathcal{M}^+$, $*$-form implication and finite implication of word constraints are also decidable in PTIME.

Proposition 8.16: Over any schema Δ in $\mathcal{M}^+$, the $*$-form implication and finite impli-

cation problems for $P_w(\Delta)$ are finitely axiomatizable and are decidable in PTIME. ∎

Indeed, it is easy to verify that the proof of Proposition 8.10 is also applicable here.

8.3 In the context of $\mathcal{M}$

In contrast to Theorems 8.1 and 8.13, this section shows that in the context of $\mathcal{M}$, path constraint implication is not only decidable in cubic-time, but is also finitely axiomatizable.

Theorem 8.17: In the context of $\mathcal{M}$, the implication and finite implication problems for path constraints (P_c) are finitely axiomatizable and are decidable in cubic-time. ∎

We first present a finite axiomatization for implication and finite implication of path constraints, and then provide a cubic-time algorithm for testing path constraint implication.

8.3.1 A finite axiomatization

We consider a set of inference rules, $\mathcal{I}_r$, consisting of the following rules and Reflexivity, Transitivity and Right-congruence given in Section 6.1:

- Commutativity:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \beta(r,x))}{\forall x\ (\beta(r,x) \rightarrow \alpha(r,x))}$$

- Forward-to-word:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(x,y)))}{\forall x\ (\alpha \cdot \beta(r,x) \rightarrow \alpha \cdot \gamma(r,x))}$$

- Word-to-forward:

$$\frac{\forall x\ (\alpha \cdot \beta(r,x) \rightarrow \alpha \cdot \gamma(r,x))}{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(x,y)))}$$

- Backward-to-word:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(y,x)))}{\forall x\ (\alpha(r,x) \rightarrow \alpha \cdot \beta \cdot \gamma(r,x))}$$

- Word-to-backward:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \alpha \cdot \beta \cdot \gamma(r,x))}{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(y,x)))}$$

Let Δ be a schema in $\mathcal{M}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_c(\Delta)$. We use $\Sigma \vdash_{\mathcal{I}_r} \varphi$ to denote that φ is provable from Σ using $\mathcal{I}_r$.

The theorem below shows that $\mathcal{I}_r$ is indeed a finite axiomatization of path constraints.

Theorem 8.18: Let Δ be any schema in $\mathcal{M}$. For every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$,

$$\Sigma \models_{\Delta} \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_r} \varphi,$$

$$\Sigma \models_{(f,\Delta)} \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_r} \varphi.$$

■

As an immediate result, over any schema Δ in $\mathcal{M}$, the implication and finite implication problems for $P_c(\Delta)$ coincide and are decidable. To see this, let $\Sigma \cup \{\varphi\}$ be any finite subset of $P_c(\Delta)$. If $\Sigma \models_{\Delta} \varphi$, then obviously $\Sigma \models_{(f,\Delta)} \varphi$. Conversely, if $\Sigma \models_{(f,\Delta)} \varphi$, then $\Sigma \vdash_{\mathcal{I}_r} \varphi$. By the soundness of $\mathcal{I}_r$ for implication, we have $\Sigma \models_{\Delta} \varphi$. Thus these two problems coincide. In addition, since $\mathcal{U}(\Delta)$ is definable in first-order logic, the equivalence of these problems leads to the decidability of both problems.

The collapse of the undecidability is due to the following lemma, which can be proved by a straightforward induction on the length of α and by using $\Phi(\Delta)$. On untyped data and in the context of $\mathcal{M}^+$ and $\mathcal{M}^+_f$, this lemma does not hold in general.

Lemma 8.19: Let Δ be an arbitrary schema in $\mathcal{M}$, and $G \in \mathcal{U}(\Delta)$. Then for every α in $Paths(\Delta)$, there is a unique $o \in |G|$, such that $G \models \alpha(r^G, o)$. ■

Using Lemma 8.19, it is easy to verify the following:

Lemma 8.20: Let Δ be a schema in $\mathcal{M}$, φ be a forward constraint of $P_c(\Delta)$:

$$\varphi = \forall\, x\, (\alpha(r,\ x) \rightarrow \forall\, y\, (\beta(x,\ y) \rightarrow \gamma(x,\ y))),$$

and ψ be a word constraint in $P_c(\Delta)$:

$$\psi = \forall\, x\, (\alpha \cdot \beta(r,\ x) \rightarrow \alpha \cdot \gamma(r,\ x)).$$

Then for any $G \in \mathcal{U}(\Delta)$, $G \models \varphi$ iff $G \models \psi$. ∎

Proof: If $G \models \neg\psi$, then there is $b \in |G|$ such that

$$G \models \alpha \cdot \beta(r^G, b) \wedge \neg\alpha \cdot \gamma(r^G, b).$$

Thus there exists $a \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. In addition, $G \models \neg\gamma(a, b)$ since otherwise $G \models \alpha \cdot \gamma(r^G, b)$. Hence there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(a, b).$$

Thus $G \models \neg\varphi$.

Conversely, if $G \models \neg\varphi$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(a, b).$$

By Lemma 8.19, a is the unique node such that $G \models \alpha(r^G, a)$. Thus

$$G \models \alpha \cdot \beta(r^G, b) \wedge \neg\alpha \cdot \gamma(r^G, b).$$

That is, $G \models \neg\psi$. ∎

Lemma 8.21: Let Δ be a schema in $\mathcal{M}$, φ be a backward constraint of $P_c(\Delta)$:

$$\varphi = \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))),$$

and ψ be a word constraint in $P_c(\Delta)$:

$$\psi = \forall x\,(\alpha(r, x) \rightarrow \alpha \cdot \beta \cdot \gamma(r, x)).$$

Then for any $G \in \mathcal{U}(\Delta)$, $G \models \varphi$ iff $G \models \psi$. ∎

Proof: If $G \models \neg\psi$, then there is $a \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \neg\alpha \cdot \beta \cdot \gamma(r^G, a).$$

By Lemma 8.19, there is $b \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. Clearly, $G \models \neg\gamma(b, a)$ since otherwise $G \models \alpha \cdot \beta \cdot \gamma(r^G, a)$. Hence there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(b, a).$$

Thus $G \models \neg\varphi$.

Conversely, if $G \models \neg\varphi$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(b,\, a).$$

By Lemma 8.19, a is the unique node such that $G \models \alpha(r^G,\, a)$, and b is the unique node such that $G \models \beta(a,\, b)$. Therefore, $G \models \neg\alpha \cdot \beta \cdot \gamma(r^G,\, a)$ since otherwise $G \models \gamma(b,\, a)$. Hence

$$G \models \alpha(r^G,\, a) \wedge \neg\alpha \cdot \beta \cdot \gamma(r^G,\, a).$$

Thus $G \models \neg\psi$. ∎

Using these lemmas, we show Theorem 8.18 as follows.

Proof of Theorem 8.18: Soundness of $\mathcal{I}_r$ can be verified by induction on the lengths of $\mathcal{I}_r$-proofs. For the proof of completeness, it suffices to show

Claim 1: Let Δ be a schema in $\mathcal{M}$, $\Sigma \cup \{\varphi\}$ be any finite subset of $P_c(\Delta)$ and k be any natural number such that

$$k \geq max\{\, |pf(\psi)| + |lt(\psi)| + |rt(\psi)| \;\;|\;\; \psi \in \Sigma \cup \{\varphi\}\}.$$

Then there is $G \in \mathcal{U}(\Delta)$ such that

1. $G \models \Sigma$,
2. for every path ρ such that $|\rho| \leq k - |pf(\varphi) \cdot lt(\varphi)|$,
   - if $G \models \forall x\,(pf(\varphi)(r,\, x) \rightarrow \forall y\,(lt(\varphi)(x,\, y) \rightarrow \rho(x,\, y)))$, then
     $$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(pf(\varphi)(r,\, x) \rightarrow \forall y\,(lt(\varphi)(x,\, y) \rightarrow \rho(x,\, y)));$$
   - if $G \models \forall x\,(pf(\varphi)(r,\, x) \rightarrow \forall y\,(lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x)))$, then
     $$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(pf(\varphi)(r,\, x) \rightarrow \forall y\,(lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x))).$$

For if Claim 1 holds and $\Sigma \models_{\Delta} \varphi$, then by $G \models \Sigma$, we have $G \models \varphi$. In addition, by $G \in \mathcal{U}_f(\Delta)$, if $\Sigma \models_{(f,\,\Delta)} \varphi$, then we also have $G \models \varphi$. Thus again by Claim 1, $\Sigma \vdash_{\mathcal{I}_r} \varphi$.

We next show Claim 1.

Let $\Delta = (\mathcal{C},\, \nu,\, DBtype)$. We first assume that for each base type $b \in T(\Delta)$, the domain of b is infinite.

We define the structure G described in Claim 1 in two steps: we first define the k-neighborhood of G, G_k, and then construct G from G_k. The notion of k-neighborhood is described in the proof of Lemma 8.8.

To construct G_k, we define the following, which are similar to those defined in the proof of Lemma 8.11.

- $Paths^k(\Delta) = \{\alpha \mid \alpha \in Paths(\Delta),\ |\alpha| \leq k\}$.
- An equivalence relation $\approx$ on $Paths^k(\Delta)$ defined by

  $$\alpha \approx \beta \ \text{ iff } \ \Sigma \vdash_{\mathcal{I}_r} \forall x\ (\alpha(r,\, x) \rightarrow \beta(r,\, x)).$$

  It should be noted that by Commutativity in $\mathcal{I}_r$, $\Sigma \vdash_{\mathcal{I}_r} \forall x\ (\alpha(r,\, x) \rightarrow \beta(r,\, x))$ iff $\Sigma \vdash_{\mathcal{I}_r} \forall x\ (\beta(r,\, x) \rightarrow \alpha(r,\, x))$.
- $\widehat{\alpha}$ denoting the equivalence class of α with respect to $\approx$.
- $\mathcal{A} = \{\widehat{\alpha} \mid \alpha \in Paths^k(\Delta)\}$.
- $type(\widehat{\alpha}) = type(\alpha)$, where $type(\alpha)$ is the type of path α determined by Δ. This is well-defined since if α and β are in the same equivalence class, then by the definition of $P_w(\Delta)$, $type(\alpha) = type(\beta)$.

Using these notions, we define $G_k = (|G_k|, r^{G_k}, E^{G_k}, R^{G_k})$ as follows.

- For each $\widehat{\alpha} \in \mathcal{A}$, let $o(\widehat{\alpha})$ be a distinct node and let $|G_k| = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A}\}$.
- Let $r^{G_k} = o(\widehat{\epsilon})$.
- For each $\tau \in T(\Delta)$, let $R_{\tau}^{G_k} = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A},\ type(\widehat{\alpha}) = \tau\}$.

- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$, then for each $i \in [1, n]$, let $G_k \models l_i(o(\widehat{\alpha}), o(\widehat{\beta \cdot l_i}))$. Note that this is well-defined by Transitivity and Right-congruence in $\mathcal{I}_r$.

Based on G_k, we define G as follows. For each $\tau \in T(\Delta)$, let $o(\tau)$ be a distinct node. Let $G = (|G|, r^G, E^G, R^G)$, where

- $|G| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta)\}$;
- $r^G = r^{G_k}$;
- for each $\tau \in T(\Delta)$, $R^G_\tau = R^{G_k}_\tau \cup \{o(\tau)\}$;
- for each $l \in E(\Delta)$, if $G_k \models l(o, o')$, then $G \models l(o, o')$. Moreover,
  - for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and for some $i \in [1, n]$, $o(\widehat{\alpha})$ does not have any outgoing edge labeled with l_i, then let $G \models l_i(o(\widehat{\alpha}), o(\tau_i))$;
  - for each type $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or τ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, let $G \models l_i(o(\tau), o(\tau_i))$.

We now show that G is indeed the structure described in Claim 1.

1. $G \in \mathcal{U}_f(\Delta)$.

As in the proof of Lemma 8.11, it is easy to verify that $|G|$ is finite and $G \models \Phi(\Delta)$.

2. $G \models \Sigma$.

We first show the following claim.

Claim 2: For every $\alpha \in Paths^k(\Delta)$, $G \models \alpha(r^G, o(\widehat{\alpha}))$.

As an immediate result of Claim 2 and Lemma 8.19, $o(\widehat{\alpha})$ is the unique node in G such that $G \models \alpha(r^G, o(\widehat{\alpha}))$.

We show Claim 2 by induction on $|\alpha|$.

Base case: $\alpha = \epsilon$.

Recall that $r^G = o(\widehat{\epsilon})$. Obviously, $G \models \epsilon(r^G,\, o(\widehat{\epsilon}))$.

Inductive step: Assume Claim 2 for α. We next show that Claim 2 also holds for $\alpha \cdot l$, where $\alpha \cdot l \in Paths^k(\Delta)$.

By the induction hypothesis, $G \models \alpha(r^G,\, o(\widehat{\alpha}))$. Since $\alpha \cdot l \in Paths^k(\Delta)$, $|\alpha \cdot l| \leq k$. Hence $|\alpha| < k$. By $\alpha \in \widehat{\alpha}$ and the definition of G, we have

$$G \models \alpha(r^G,\, o(\widehat{\alpha})) \wedge l(o(\widehat{\alpha}),\, o(\widehat{\alpha \cdot l})).$$

That is, $G \models \alpha \cdot l(r^G,\, o(\widehat{\alpha \cdot l}))$.

Hence Claim 2 holds.

Using Claim 2, we show $G \models \Sigma$.

Suppose, for *reductio*, that there is $\psi \in \Sigma$ such that $G \models \neg\psi$. Then we show that the assumption leads to a contradiction.

If ψ is a forward constraint $\forall x\,(\alpha(r,\, x) \rightarrow \forall y\,(\beta(x,\, y) \rightarrow \gamma(x,\, y)))$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(a,\, b).$$

Thus by Lemma 8.19 and Claim 2, we have $a = o(\widehat{\alpha})$ and $b = o(\widehat{\alpha \cdot \beta})$. By Forward-to-word in $\mathcal{I}_r$, we have

$$\alpha \cdot \beta \approx \alpha \cdot \gamma.$$

Therefore, again by Claim 2, we have $G \models \alpha \cdot \gamma(r^G,\, o(\widehat{\alpha \cdot \beta}))$. By Lemma 8.19, we have

$$G \models \gamma(o(\widehat{\alpha}),\, o(\widehat{\alpha \cdot \beta})).$$

This contradicts the assumption.

If ψ is a backward constraint $\forall x\,(\alpha(r,\, x) \rightarrow \forall y\,(\beta(x,\, y) \rightarrow \gamma(y,\, x)))$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(b,\, a).$$

Again by Lemma 8.19 and Claim 2, we have $a = o(\widehat{\alpha})$ and $b = o(\widehat{\alpha \cdot \beta})$. By Backward-to-word in $\mathcal{I}_r$, we have

$$\alpha \approx \alpha \cdot \beta \cdot \gamma.$$

Therefore, again by Claim 2, we have $G \models \alpha \cdot \beta \cdot \gamma(r^G,\, o(\widehat{\alpha}))$. By Lemma 8.19, we have

$$G \models \gamma(o(\widehat{\alpha \cdot \beta}),\, o(\widehat{\alpha})).$$

This again contradicts the assumption.

Thus $G \models \psi$. Hence $G \models \Sigma$.

3. G has the property described by (2) of Claim 1.

Let ρ be a path such that $|\rho| \leq k - |pf(\varphi) \cdot lt(\varphi)|$.

If $G \models \forall x\, (pf(\varphi)(r,\, x) \to \forall y\, (lt(\varphi)(x,\, y) \to \rho(x,\, y)))$, then by Lemma 8.20,

$$G \models \forall x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x) \to pf(\varphi) \cdot \rho(r,\, x)).$$

By Claim 2 and Lemma 8.19, we have

$$G \models pf(\varphi) \cdot \rho(r^G,\, o(\widehat{pf(\varphi) \cdot lt(\varphi)})),$$

and moreover,

$$pf(\varphi) \cdot lt(\varphi) \approx pf(\varphi) \cdot \rho.$$

Thus by the definition of $\approx$ and Commutativity in $\mathcal{I}_r$, we have

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x) \to pf(\varphi) \cdot \rho(r,\, x)).$$

By Word-to-forward in $\mathcal{I}_r$, we have

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\, (pf(\varphi)(r,\, x) \to \forall y\, (lt(\varphi)(x,\, y) \to \rho(x,\, y))).$$

If $G \models \forall x\, (pf(\varphi)(r,\, x) \to \forall y\, (lt(\varphi)(x,\, y) \to \rho(y,\, x)))$, then by Lemma 8.21, we have

$$G \models \forall x\, (pf(\varphi)(r,\, x) \to pf(\varphi) \cdot lt(\varphi) \cdot \rho(r,\, x)).$$

By Claim 2 and lemma 8.19, we have

$$G \models pf(\varphi) \cdot lt(\varphi) \cdot \rho(r^G,\, o(\widehat{pf(\varphi)})),$$

and moreover,

$$pf(\varphi) \approx pf(\varphi) \cdot lt(\varphi) \cdot \rho.$$

Thus by the definition of $\approx$ and Commutativity in $\mathcal{I}_r$, we have

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\, (pf(\varphi)(r,\, x) \rightarrow pf(\varphi) \cdot lt(\varphi) \cdot \rho(r,\, x)).$$

By Word-to-backward in $\mathcal{I}_r$, we have

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\, (pf(\varphi)(r,\, x) \rightarrow \forall y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x))).$$

Thus when all the base types in $T(\Delta)$ have infinite domains, Claim 1 holds. As in the proof of Lemma 8.11, when some base types in $T(\Delta)$ have finite domains, the argument above can be modified slightly to establish Claim 1 in this case.

This completes the proof of Claim 1, and therefore, the proof of Theorem 8.18. ∎

8.3.2 An algorithm

Next, we present an algorithm for testing path constraint implication in the context of $\mathcal{M}$. Let Δ be a schema in $\mathcal{M}$. Similar to Algorithm 6.2, this algorithm takes as input a finite subset Σ of $P_c(\Delta)$ and a path $\alpha \cdot \beta$ in $Paths(\Delta)$. It computes a pseudo model G of Σ having the following properties: $G \models \Sigma$ and there are $a, b \in |G|$ such that $G \models \alpha(r^G,\, a) \wedge \beta(a,\, b)$. In addition, for any path γ,

$$\begin{aligned} G &\models \gamma(a,\, b) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\, (\alpha(r,\, x) \rightarrow \forall y\, (\beta(x,\, y) \rightarrow \gamma(x,\, y))), \\ G &\models \gamma(b,\, a) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\, (\alpha(r,\, x) \rightarrow \forall y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x))). \end{aligned}$$

By Theorem 8.18, this algorithm can be used for testing implication and finite implication of constraints of $P_c(\Delta)$.

Before we present the algorithm, we first define the following. Let Δ be a schema in $\mathcal{M}$, Σ be a finite subset of $P_c(\Delta)$ and $\alpha \cdot \beta \in Paths(\Delta)$. We define

$$\begin{aligned} Pts(\Sigma,\, \alpha \cdot \beta) &= \{\alpha \cdot \beta\} \cup \\ &\quad \{pf(\psi) \cdot lt(\psi),\ pf(\psi) \cdot rt(\psi) \mid \psi \in \Sigma,\ \psi \text{ is of the forward form}\} \cup \\ &\quad \{pf(\psi) \cdot lt(\psi) \cdot rt(\psi) \mid \psi \in \Sigma,\ \psi \text{ is of the backward form}\}, \\ CloPts(\Sigma,\, \alpha \cdot \beta) &= \{\rho \mid \varrho \in Pts(\phi),\ \rho \preceq_p \varrho\}. \end{aligned}$$

Here $\rho \preceq_p \varrho$ denotes that ρ is a prefix of ϱ.

Using these notions, we give the algorithm (Algorithm 8.1) in Table 8.1. The procedure $merge(a,\, b)$ used in Algorithm 8.1 is given in Table 8.2.

The following should be noted about Algorithm 8.1.

Remark 1: Algorithm 8.1 is independent of any particular schema. Although it is required that input constraints and path are defined over some schema in $\mathcal{M}$, no particular information about the schema is used by the algorithm. As a result, this algorithm can be used in the context of any schema in $\mathcal{M}$.

Remark 2: Let Δ be a schema in $\mathcal{M}$, and an input of the algorithm be a finite subset Σ of $P_c(\Delta)$ and a path $\alpha \cdot \beta \in Paths(\Delta)$. Then the structure G computed by the algorithm may not be in $\mathcal{U}(\Delta)$. However, G can be naturally extended to a structure $H \in \mathcal{U}_f(\Delta)$, as follows. Let $H = (|H|,\, r^H,\, E^H,\, R^H)$, where

- $|H| = |G| \cup \{o(\tau) \mid \tau \in T(\Delta),\ o(\tau) \text{ is a distinct node}\}$;
- $r^H = r^G$;
- for each $\tau \in T(\Delta)$,

$$R^H_\tau = \{o(\tau)\} \cup \{o \mid o \in |G|,\ \rho \in Paths(\Delta),\ type(\rho) = \tau,\ G \models \rho(r^G,\, o)\};$$

- for each $l \in E(\Delta)$, if $G \models l(o,\, o')$, then $H \models l(o,\, o')$. Moreover,
  - for each $o \in |G|$, if there exists path $\rho \in Paths(\Delta)$ such that $G \models \rho(r^G,\, o)$ and $type(\rho) = [l_1 : \tau_1,\, ...,\, l_n : \tau_n]$ (or $type(\rho)$ is some class C in $\mathcal{C}$ and moreover,

Algorithm 8.1:

Input: a finite subset Σ of $P_c(\Delta)$ and a path $\alpha \cdot \beta \in Paths(\Delta)$
Output: the structure G described above

1. E_ϕ := the set of edge labels appearing in either $\alpha \cdot \beta$ or some path in constraints of Σ;

2. $Rules := \Sigma$;

3. $G := (|G|, r^G, E_\phi^G)$, where
   - $|G| = \{o(\rho) \mid \rho \in CloPts(\Sigma, \alpha \cdot \beta),\ o(\rho)$ is a distinct node$\}$,
   - $r^G = o(\epsilon)$,
   - E_ϕ^G is populated such that $G \models l(o(\rho), o(\varrho))$ iff $\varrho = \rho \cdot l$;

4. for each $\psi \in \Sigma$ do:
   (1) if $\psi = \forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(x, y)))$ then
      (i) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(x, y)))\}$;
      (ii) $merge(o_{\rho\cdot\varrho}, o_{\rho\cdot\zeta})$,
         where $o_{\rho\cdot\varrho}, o_{\rho\cdot\zeta} \in |G|$ such that $G \models \rho \cdot \varrho(r^G, o_{\rho\cdot\varrho}) \wedge \rho \cdot \zeta(r^G, o_{\rho\cdot\zeta})$;
   (2) if $\psi = \forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(y, x)))$ then
      (i) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \rightarrow \forall y\,(\varrho(x, y) \rightarrow \zeta(y, x)))\}$;
      (ii) $merge(o_\rho, o_{\rho\cdot\varrho\cdot\zeta})$,
         where $o_\rho, o_{\rho\cdot\varrho\cdot\zeta} \in |G|$ such that $G \models \rho(r^G, o_\rho) \wedge \rho \cdot \varrho \cdot \zeta(r^G, o_{\rho\cdot\varrho\cdot\zeta})$;

5. output G.

Table 8.1: An algorithm for testing path constraint implication in $\mathcal{M}$

procedure $merge(a, b)$

1. for each $o \in |G|$ do
   if there is $l \in E_\phi^G$ such that $G \models l(o, b)$ then
      (1) delete from E_ϕ^G the edge labeled l from o to b;
      (2) add to E_ϕ^G an edge labeled l from o to a;
2. for each $l \in E_\phi$ do
   if there are $o_a, o_b \in |G|$ such that $G \models l(b, o_b) \wedge l(a, o_a)$ and $o_a \neq o_b$ then
      (1) delete from E_ϕ^G the edge labeled l from b to o_b;
      (2) add to E_ϕ^G an edge labeled l from a to o_b;
      (3) $merge(o_a, o_b)$;
3. $|G| := |G| \setminus \{b\}$;

Table 8.2: Procedure *merge* used in Algorithm 8.1

$\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and in addition, for some $i \in [1, n]$, o does not have an outgoing edge labeled l_i, then let $H \models l_i(o, o(\tau_i))$;

- for each type $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or τ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, let $H \models l_i(o(\tau), o(\tau_i))$.

It is easy to verify that $H \in \mathcal{U}(\Delta)$ as long as $\Sigma \subseteq P_c(\Delta)$ and $\alpha \cdot \beta \in Paths(\Delta)$. In addition, it is easy to verify that H is finite, and therefore, $H \in \mathcal{U}_f(\Delta)$.

We call the structure H defined above *the extension of G with respect to Δ*.

Remark 3: The rationale behind the procedure *merge* is Commutativity, Transitivity and Right-congruence in $\mathcal{I}_r$.

Remark 4: The rationale behind step 4 (1) (i) and 4 (2) (i) of Algorithm 8.1 is Lemma 8.19. Let Δ be a schema in $\mathcal{M}$ and $G \in \mathcal{U}(\Delta)$. For any path $\rho \in Paths(\Delta)$, there exists a unique $o \in |G|$ such that $G \models \rho(r^G, o)$. As a result, every constraint in Σ is applied only once by the algorithm. It is because of this property that Algorithm 8.1 has low complexity.

Next, we analyze the complexity of the algorithm. Let n_E be the cardinality of E_ϕ, n_C the cardinality of $CloPts(\Sigma, \alpha \cdot \beta)$, n_G the size of $|G|$, n the length of Σ and $\alpha \cdot \beta$, and n_Σ the cardinality of Σ. Then the following should be noted.

- $n_E \leq n$, $n_C \leq n$, $n_G \leq n$ and $n_\Sigma \leq n$.
- Step 4 is executed n_Σ times.
- Testing whether $G \models \rho(r^G, o_\rho)$ in step 4 can be done in at most $O(n_G\,|\rho|)$ time. Therefore, it can be done in $O(n^2)$ time. By using appropriate data structure, e.g., (variable length) array indexed by edge labels in E_ϕ, this can be done in $O(|\rho|)$ time, i.e., $O(n)$ time.
- The procedure *merge* is executed at most n_G times. Each step takes $O(n_E\ n_G)$ time. Hence the total cost of executing *merge* is $O(n_G^2\, n_E)$, i.e., $O(n^3)$. Again, by using appropriate data structure, this can be done in $O(n^2)$ time.

Therefore, Algorithm 8.1 runs in $O(n^3)$ time. In addition, when implemented using appropriate data structures, this algorithm runs in $O(n^2)$ time.

The proposition below shows that Algorithm 8.1 is correct.

Proposition 8.22: Let Δ be a schema in $\mathcal{M}$. Given a finite subset Σ of $P_c(\Delta)$ and path $\alpha \cdot \beta \in Paths(\Delta)$, Algorithm 8.1 computes a structure G having the following property: $G \models \Sigma$, and there are $a, b \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. In addition, for any path γ,

$$G \models \gamma(a, b) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y))),$$
$$G \models \gamma(b, a) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))).$$

■

Proof: Step 4 of Algorithm 8.1 ensures that $G \models \Sigma$, because of Lemma 8.19. In addition, step 3 ensures that there are $a, b \in |G|$, such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. Let H denote the extension of G with respect to Δ. Then it is easy to verify that $H \models \alpha(r^G, a) \wedge \beta(a, b)$ and $H \models \Sigma$. Thus if $\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y)))$, then by Theorem 8.18, $H \models \gamma(a, b)$. By the definition of H, it is easy to verify that

$$G \models \gamma(a, b).$$

Similarly, if $\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x)))$, then $H \models \gamma(b, a)$. Again by the definition of H, it is easy to verify that

$$G \models \gamma(b, a).$$

Conversely, by a straightforward induction on the number of steps in the construction of G by the algorithm, it can be shown that for all paths ρ and ϱ, if there exists a node $o \in |G|$ such that $G \models \rho(r^G, o) \wedge \varrho(r^G, o)$, then $\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\rho(r, x) \rightarrow \varrho(r, x))$. Indeed, each step of the construction in fact corresponds to applications of some rules in $\mathcal{I}_r$. For example, step 4 (1) corresponds to an application of Forward-to-word, step 4 (2) corresponds to an application of Backward-to-word, and *merge* corresponds to applications of Transitivity,

Right-congruence and Commutativity in $\mathcal{I}_r$. As a result, if $G \models \gamma(a, b)$, then by Word-to-forward in $\mathcal{I}_r$, it can be verified that

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\ x) \rightarrow \forall y\,(\beta(x,\ y) \rightarrow \gamma(x,\ y))).$$

Similarly, if $G \models \gamma(b, a)$, then by Word-to-backward in $\mathcal{I}_r$, it can be verified that

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\ x) \rightarrow \forall y\,(\beta(x,\ y) \rightarrow \gamma(y,\ x))).$$

■

From Proposition 8.22, Algorithm 8.1 and Theorem 8.18, Theorem 8.17 follows immediately.

Chapter 9

Interaction between Path and Type Constraints

This chapter investigates the interaction between path constraints and type constraints. It shows that in general, results on path constraint implication developed in the context of semistructured data may no longer hold when a type is imposed on the data. The presence of types may in some cases simplify reasoning about path constraints, and in other cases make it harder. To demonstrate this, two fragments of the path constraint language P_c and their associated implication and finite implication problems are investigated:

- Two restricted implication problems associated with path constraints of P_c are identified in Section 9.1, referred to as EIPs (extended implication and finite implication problems for word constraints) and PBIPs (prefix bounded implication and finite implication problems for path constraints), respectively.
- On the one hand, Section 9.2 shows that in the context of the semistructured data model $\mathcal{SM}$, EIPs are undecidable, but in the context of the object-oriented model $\mathcal{M}$, these problems become decidable in cubic-time.
- On the other hand, Section 9.3 shows that in $\mathcal{SM}$, PBIPs are decidable in PTIME, but in the context of the object-oriented models $\mathcal{M}^+$ and $\mathcal{M}_f^+$, these problems become undecidable.

9.1 Restricted implication problems: EIPs and PBIPs

The aim of Chapter 9 is to demonstrate the impact of type constraints on path constraint implication. In the last chapter, we have shown that some undecidability results established in the context of semistructured data break down in the presence of types. More specifically, the implication and finite implication problems for P_c are undecidable in the

semistructured data model $\mathcal{SM}$ (Theorem 4.1). However, although these problems remain undecidable in the context of the object-oriented models $\mathcal{M}_f^+$ and $\mathcal{M}^+$ (Theorems 8.1 and 8.13), they become decidable in cubic-time in the object-oriented model $\mathcal{M}$ (Theorem 8.17). In this chapter, we show that these results also hold on a fragment of P_c, whose associated implication and finite implication problems are referred to as EIPs (extended implication and finite implication problems for P_w).

One is tempted to think that adding type to data simplifies reasoning about path constraints, as demonstrated above. However, this is not always the case. In this chapter, we show that some decidability results established in the context of semistructured data also break down when a type is imposed on the data. More specifically, we identify another fragment of P_c, whose associated implication and finite implication problems are referred to as PBIPs (prefix bounded implication and finite implication problems for P_c). We show that PBIPs are decidable in PTIME in the semistructured data model $\mathcal{SM}$, but become undecidable in the object-oriented models $\mathcal{M}_f^+$ and $\mathcal{M}^+$.

This section describes these two fragments and their associated implication and finite implication problems.

9.1.1 The extended implication problems for P_w

Recall the class of word constraints, P_w, described in Chapter 2, and the prefix extension function δ defined in Definition 5.3. Given a constraint $\varphi \in P_w$:

$$\forall\, x\, (\beta(r,\, x) \rightarrow \gamma(r,\, x))$$

and a path α, the function δ *extends* φ *with prefix* α as follows:

$$\delta(\varphi,\, \alpha) = \forall\, x\, (\alpha(r, x) \rightarrow \forall\, y\, (\beta(x, y) \rightarrow \gamma(x, y))).$$

Using P_w and δ, below we define a fragment of P_c:

Definition 9.1: Let α be a path. The *extension of* P_w *with prefix* α is defined to be

$$P_w(\alpha) = P_w \cup \{\delta(\varphi,\, \alpha) \mid \varphi \in P_w\}.$$

We use P_w^α to denote the set $\{\delta(\varphi, \alpha) \mid \varphi \in P_w\}$. ∎

Obviously, $P_w(\alpha)$ is a mild generalization of P_w.

The driving force behind the interest in $P_w(\alpha)$ is its ability to express extent and local extent constraints. As demonstrated in Chapter 1, P_w is capable of expressing extent constraints, which are important integrity constraints commonly arise in practice. As illustrated in Section 5.1, in general, when represented in a global environment, extent constraints in a local database are augmented with some common prefix α. These local extent constraints are represented as constraints in P_w^α.

As an example, consider the XML document given in Chapter 1. Suppose this document is the bibliography of the library at University of Pennsylvania. This bibliography can be viewed as a database, as depicted in Figure 1.2. Let us refer to this database as Penn-bib. The following constraints of P_w express extent constraints on Penn-bib:

$$\begin{aligned} \varphi_1 &= \forall x\,(book \cdot author(r,\, x) \rightarrow person(r,\, x)) \\ \varphi_2 &= \forall x\,(person \cdot wrote(r,\, x) \rightarrow book(r,\, x)) \\ \varphi &= \forall x\,(book \cdot author \cdot wrote(r,\, x) \rightarrow person(r,\, x)) \end{aligned}$$

It is natural to expect that Penn-bib has links to external resources, e.g., bibliographies at MIT, Warner and O'Reilly. In XLink (XML Link language [63]), these links can be specified by:

```
<external xml:  link = "external"   inline = "false" >
          <locator   href = "MIT" />
          <locator   href = "Warner" />
          <locator   href = "O'Reilly" />
</external>
```

Each of these external bibliographies can also be treated as a database of the form depicted in Figure 1.2. These databases are *local databases* of Penn-bib. In our graph representation, there are edges emanating from the root node of Penn-bib that are labeled with `MIT`, `Warner`, `O'Reilly`, and lead to these local databases. It is natural to expect the extent constraints above to hold on these local databases. For example, we want the following

constraints to hold on the MIT database:

$$\begin{aligned}
\phi_1 &= \forall x\,(MIT(r,\,x) \rightarrow \forall y\,(book \cdot author(x,\,y) \rightarrow person(x,\,y))) \\
\phi_2 &= \forall x\,(MIT(r,\,x) \rightarrow \forall y\,(person \cdot wrote(x,\,y) \rightarrow book(x,\,y))) \\
\phi &= \forall x\,(MIT(r,\,x) \rightarrow \forall y\,(book \cdot author \cdot wrote(x,\,y) \rightarrow person(x,\,y)))
\end{aligned}$$

These constraints are called *local extent constraints*. It should be noted that φ_1, φ_2, φ, ϕ_1, ϕ_2 and ϕ are in $P_w(MIT)$.

Next, we describe the implication and finite implication problems associated with $P_w(\alpha)$.

Definition 9.2: In the context of the semistructured data model $\mathcal{SM}$, the *extended (finite) implication problem for word constraints (P_w)* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\alpha)$, where α is a path, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$), i.e., whether all the (finite) σ-structures that satisfy Σ are also models of φ.

We use the abbreviation *EIPs* to refer to these problems. ∎

For example, let Σ be the set consisting of φ_1, φ_2, ϕ_1, ϕ_2 and ϕ given above. The question whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) is an instance of the extended (finite) implication problem for P_w.

In the context of structured data, EIPs can be defined similarly.

Definition 9.3: Let Δ be a schema in $\mathcal{M}_f^+$ and α be a path in $Paths(\Delta)$. The *extension of $P_w(\Delta)$ with prefix α* is defined to be

$$P_w(\Delta,\,\alpha) = P_w(\Delta) \cup \{\delta(\varphi,\,\alpha) \mid \varphi \in P_w(\Delta),\ \delta(\varphi,\,\alpha) \in P_c(\Delta)\}.$$

We use $P_w^\alpha(\Delta)$ to denote the set $\{\delta(\varphi,\,\alpha) \mid \varphi \in P_w(\Delta),\ \delta(\varphi,\,\alpha) \in P_c(\Delta)\}$. ∎

Definition 9.4: In the context of $\mathcal{M}_f^+$, the *extended (finite) implication problem for word constraints (P_w)* is the problem of determining, given any schema Δ in $\mathcal{M}_f^+$ and any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta,\,\alpha)$, where $\alpha \in Paths(\Delta)$, whether $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\,\Delta)} \varphi$), i.e., whether for any $G \in \mathcal{U}(\Delta)$ ($G \in \mathcal{U}_f(\Delta)$), if $G \models \Sigma$ then $G \models \varphi$. ∎

In the same way, $P_w(\Delta,\,\alpha)$ and EIPs can be defined in the context of $\mathcal{M}^+$ and $\mathcal{M}$.

9.1.2 The prefix bounded implication problems for P_c

Next, we introduce another fragment of P_c.

Definition 9.5: Let α be a path and K a binary relation symbol. A constraint φ of P_c is said to be *bounded by α and K* if it is of the form

$$\forall x\,(\alpha \cdot K(r,x) \rightarrow \forall y\,(\beta(x,y) \rightarrow \gamma(x,y))),$$

where $\beta \neq \epsilon$ and $K \not\preceq_p \beta$ (i.e., K is not a prefix of β).

A *subset Σ of P_c with prefix bounded by α and K* is a finite subset of P_c such that for each $\varphi \in \Sigma$, either φ is bounded by α and K, or for some path α', $pf(\varphi) = \alpha \cdot \alpha'$ and $K \not\preceq_p \alpha'$. In addition, if $\alpha' = \epsilon$, then φ is of the form

$$\forall x\,(\alpha(r,x) \rightarrow \forall y\,(\epsilon(x,y) \rightarrow K(x,y))).$$

■

Definition 9.6: In the context of the semistructured data model $\mathcal{SM}$, the *prefix bounded (finite) implication problem for path constraints (P_c)* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of P_c with prefix bounded by α and K, where α is a path, K is a binary relation symbol and φ is bounded by α and K, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$).

We use the abbreviation *PBIPs* to refer to these problems. ■

As an example, consider the set consisting of ϕ_1, ϕ_2, ϕ given above and the following:

$$\begin{aligned}
\psi_1 &= \forall x\,(Warner(r,\,x) \rightarrow \forall y\,(book \cdot author(x,\,y) \rightarrow person(x,\,y))) \\
\psi_2 &= \forall x\,(Warner(r,\,x) \rightarrow \forall y\,(person \cdot wrote(x,\,y) \rightarrow book(x,\,y)))
\end{aligned}$$

$$\begin{aligned}
\psi_1' &= \forall x\,(O'Reilly(r,\,x) \rightarrow \forall y\,(book \cdot author(x,\,y) \rightarrow person(x,\,y))) \\
\psi_2' &= \forall x\,(O'Reilly(r,\,x) \rightarrow \forall y\,(person \cdot wrote(x,\,y) \rightarrow book(x,\,y)))
\end{aligned}$$

Then this set can be viewed as a subset of P_c with prefix bounded by the empty path ϵ and MIT. In particular, ϕ_1, ϕ_2 and ϕ are constraints bounded by ϵ and MIT. Let Σ be

the set $\{\phi_1, \phi_2, \psi_1, \psi_2, \psi_1', \psi_2'\}$. The question whether $\Sigma \models \phi$ ($\Sigma \models_f \phi$) is an instance of the prefix bounded (finite) implication problem for P_c.

Intuitively, PBIPs describe implication and finite implication of certain local extent constraints. More specifically, let DB be a database and DB_l be a local database of DB which is connected to DB by path $\alpha \cdot K$. Constraints bounded by α and K can be viewed as local extent constraints on DB_l. A subset Σ of P_c with prefix bounded by α and K consists of such local extent constraints and constraints on other local databases connected to DB by some path $\alpha \cdot \alpha'$, where $K \not\preceq_p \alpha'$. It can be partitioned into Σ_1 and Σ_2, where Σ_1 consists of local extent constraints on DB_l, and Σ_2 contains constraints on other local databases. PBIPs study implication and finite implication of local extent constraints on DB_l (i.e., Σ_1) in the presence of constraints on other local database (i.e., Σ_2).

In the context of structured data, PBIPs can also be defined.

Definition 9.7: Let Δ be a schema in $\mathcal{M}_f^+$, $\alpha \in Paths(\Delta)$ and $K \in E(\Delta)$. A constraint φ of $P_c(\Delta)$ is said to be *bounded by α and K* if it is of the form

$$\forall x\,(\alpha \cdot K(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y))),$$

where $\beta \neq \epsilon$ and $K \not\preceq_p \beta$.

A *subset Σ of $P_c(\Delta)$ with prefix bounded by α and K* is a finite subset of $P_c(\Delta)$ such that for each $\varphi \in \Sigma$, either φ is bounded by α and K, or for some path α' over Δ, $pf(\varphi) = \alpha \cdot \alpha'$ and $K \not\preceq_p \alpha'$. In addition, if $\alpha' = \epsilon$, then φ is of the form

$$\forall x\,(\alpha(r, x) \rightarrow \forall y\,(\epsilon(x, y) \rightarrow K(x, y))).$$

■

Definition 9.8: In the context of $\mathcal{M}_f^+$, the *prefix bounded (finite) implication problem for path constraints (P_c)* is the problem of determining, given any schema Δ in $\mathcal{M}_f^+$ and any finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$ with prefix bounded by α and K, where $\alpha \in Paths(\Delta)$, $K \in E(\Delta)$ and φ is bounded by α and K, whether $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\Delta)} \varphi$). ■

In the same way, PBIPs can be defined in the context of $\mathcal{M}^+$ and $\mathcal{M}$.

9.2 Undecidability and decidability of EIPs

This section investigates EIPs (the extended implication and finite implication problems for word constraints). The main results of this section are the following.

1. In the context of the semistructured data model $\mathcal{SM}$, EIPs are undecidable.
2. In contrast, in the context of the object-oriented data model $\mathcal{M}$, EIPs are decidable in cubic-time.
3. In the context of $\mathcal{M}_f^+$, EIPs remain undecidable.
4. In the context of $\mathcal{M}^+$, these problems are also undecidable.

9.2.1 The undecidability of EIPs in the context of $\mathcal{SM}$

We begin by establishing the undecidability of the extended implication and finite implication problems for word constraints in the context of $\mathcal{SM}$.

Theorem 9.1: In the context of the semistructured data model $\mathcal{SM}$, the extended implication and finite implication problems for word constraints are undecidable. ∎

In fact, we show that these undecidability results also hold for a special case of the extended (finite) implication problem, defined as follows.

Definition 9.9: In the context of $\mathcal{SM}$, the *unit extended (finite) implication problem for word constraints* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(K)$, where $K \in E$ (E is the set of binary relation symbols in signature σ), whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$). ∎

For the unit extended (finite) implication problem for word constraints, we have the following theorem, from which Theorem 9.1 follows immediately.

Theorem 9.2: In the context of $\mathcal{SM}$, the unit extended implication and finite implication

problems for word constraints are undecidable. ∎

These theorems strengthen the undecidability results established in Chapter 4 by providing an undecidability fragment of P_c smaller than P_+ and P_f, which are the two undecidable sublanguages studied in Chapter 4.

These new undecidability results are rather surprising. As shown by [9], in $\mathcal{SM}$ the implication and finite implication problems for P_w are decidable in PTIME. As a result, by Theorem 5.11, it can be shown that for any path α, the implication and finite implication problems for P_w^α are also decidable in PTIME. However, Theorem 9.1 shows that $P_w(\alpha)$, which is $P_w \cup P_w^\alpha$, possesses undecidable implication and finite implication problems.

We prove Theorem 9.2 by reduction from the word problem for (finite) monoids. Recall the word problem described in Section 6.4. We first present an encoding of the word problem for (finite) monoids in terms of the unit extended (finite) implication problem for word constraints, and then show that the encoding is indeed a reduction.

Let $,_0$ be a finite alphabet and Θ_0 be a finite set of equations (over $,_0$). Without loss of generality, assume

$$
\begin{aligned}
,_0 &= \{l_j \mid j \in [1,m],\ l_i \neq l_j \text{ if } i \neq j\},\\
\Theta_0 &= \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in ,_0^*,\ i \in [1,n]\}.
\end{aligned}
$$

Then we define a first-order logic signature

$$\sigma_0 = (r,\ ,_0 \cup \{K\}),$$

where $K \notin ,_0$, r is a constant symbol, and $,_0 \cup \{K\}$ is a set of binary relation symbols. Note here that each symbol in $,_0$ is a binary relation symbol in σ_0. Therefore, every α in $,_0^*$ can be represented as a path formula, also denoted by α. In addition, we use $\cdot$ to denote the concatenation operator for both paths and strings.

We encode Θ_0 in terms of $\Sigma_1 \subseteq P_w$ and $\Sigma_2 \subseteq P_w^K$.

1. Σ_1 consists of the following constraints:

$$\forall x\,(\epsilon(r,x) \quad \rightarrow \quad K(r,x))$$

$$\forall\, x\, (K \cdot l_j(r, x) \quad \rightarrow \quad K(r, x))$$

for every $j \in [1, m]$.

2. Σ_2 consists of the following constraints:

$$\forall\, x\, (K(r, x) \quad \rightarrow \quad \forall\, y\, (\alpha_i(x, y) \rightarrow \beta_i(x, y)))$$
$$\forall\, x\, (K(r, x) \quad \rightarrow \quad \forall\, y\, (\beta_i(x, y) \rightarrow \alpha_i(x, y)))$$

for every $(\alpha_i,\ \beta_i) \in \Theta_0$.

Let $(\alpha,\ \beta)$ be a test equation, where α and β are arbitrary strings in $,\ {}^*_0$. We encode such a pair of strings with a pair of constraints in P_w:

$$\varphi_{(\alpha,\beta)} \quad = \quad \forall\, x\, (\alpha(r, x) \rightarrow \beta(r, x))$$
$$\varphi_{(\beta,\alpha)} \quad = \quad \forall\, x\, (\beta(r, x) \rightarrow \alpha(r, x))$$

We reduce the word problem for monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models \varphi_{(\alpha,\beta)} \wedge \varphi_{(\beta,\alpha)},$$

and analogously, reduce the word problem for finite monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models_f \varphi_{(\alpha,\beta)} \wedge \varphi_{(\beta,\alpha)}.$$

Before we show that this encoding is indeed a reduction, we first present a basic property of Σ_1.

Lemma 9.3: For every σ_0-structure G, if $G \models \Sigma_1$, then for every $\alpha \in ,\ {}^*_0$ and $o \in |G|$ such that $G \models \alpha(r^G,\ o)$, we have

$$G \models K(r^G,\ o).$$

In addition, for all $o, o' \in |G|$ such that $G \models K(r^G,\ o') \wedge \alpha(o',\ o)$, we have

$$G \models K(r^G,\ o).$$

■

Proof: By a straightforward induction on $|\alpha|$. ■

Next, we show that the encoding given above is indeed a reduction from the word problem for (finite) monoids. It suffices to show the following lemma.

Lemma 9.4: In the context of $\mathcal{SM}$, for all α and β in $,\ {}^{*}_{0}$,

$$\Theta_0 \models (\alpha,\ \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models \forall\, x\, (\alpha(r,x) \rightarrow \beta(r,x)) \ \wedge\ \forall\, x\, (\beta(r,x) \rightarrow \alpha(r,x)), \qquad \text{(a)}$$

$$\Theta_0 \models_f (\alpha,\ \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_f \forall\, x\, (\alpha(r,x) \rightarrow \beta(r,x)) \ \wedge\ \forall\, x\, (\beta(r,x) \rightarrow \alpha(r,x)). \qquad \text{(b)}$$

■

Proof: We prove (b) only. The proof of (a) is similar and simpler.

The argument is similar to the proof of Theorem 6.19.

(*if*) Suppose that $\Theta_0 \not\models_f (\alpha,\ \beta)$. Then we show that $\Sigma_1 \cup \Sigma_2 \not\models_f \forall\, x\, (\alpha(r,x) \rightarrow \beta(r,x))$. That is, we show that there exists a finite σ_0-structure G, such that $G \models \Sigma_1 \cup \Sigma_2$ but $G \not\models \forall\, x\, (\alpha(r,x) \rightarrow \beta(r,x))$.

To do this, we first define some notations. By $\Theta_0 \not\models_f (\alpha,\ \beta)$, there exist a finite monoid $(M,\ \circ,\ 1)$ and a homomorphism $h : ,\ {}^{*}_{0} \rightarrow M$ such that for every $i \in [1,n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Based on M and h, we define an equivalence relation $\approx$ on $,\ {}^{*}_{0}$ as follows:

$$\rho \approx \varrho \quad \text{iff} \quad h(\rho) = h(\varrho).$$

For every $\rho \in ,\ {}^{*}_{0}$, let $\widehat{\rho}$ be the equivalence class of ρ with respect to $\approx$. Let

$$C_{\Theta_0} = \{\widehat{\rho} \mid \rho \in ,\ {}^{*}_{0}\}.$$

Using these notations, we construct a structure $G = (|G|,\ r^G,\ E^G)$ as follows.

(1) $|G|$.

For each $\widehat{\rho} \in C_{\Theta_0}$, let $o(\widehat{\rho})$ be a distinct node. Then we define

$$|G| = \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}.$$

(2) $r^G = o(\widehat{\epsilon})$.

(3) The binary relations are populated as follows: For each $\widehat{\rho} \in C_{\Theta_0}$, let $G \models K(o(\widehat{\epsilon}),\ o(\widehat{\rho}))$. In addition, for each $j \in [1,m]$, let $G \models l_j(o(\widehat{\rho}),\ o(\widehat{\rho \cdot l_j}))$.

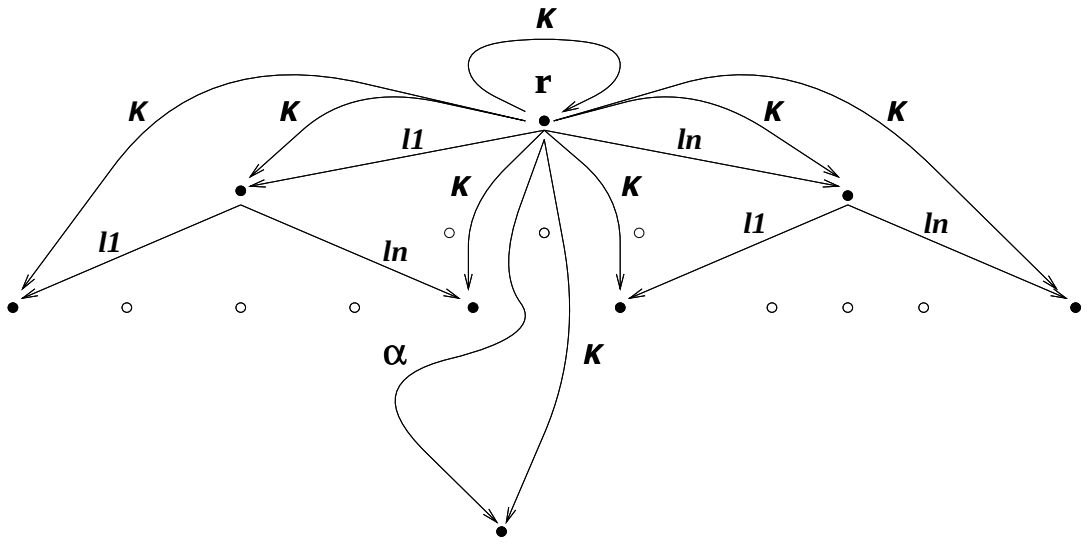

Figure 9.1: The structure G in the proof of Lemma 9.4

The structure G is shown in Figure 9.1. Obviously, G is not a deterministic structure.

By the construction of G, it is easy to see that for every $\rho \in {}^{*}_{0}$ and $j \in [1, m]$, $o(\widehat{\rho \cdot l_j})$ is the unique node such that $G \models l_j(o(\widehat{\rho}), o(\widehat{\rho \cdot l_j}))$. This is because h is a homomorphism, and as a result, if $\rho_1 \approx \rho_2$, then

$$\begin{aligned} h(\rho_1 \cdot l_j) &= h(\rho_1) \circ h(l_j) \\ &= h(\rho_2) \circ h(l_j) \\ &= h(\rho_2 \cdot l_j). \end{aligned}$$

Using this property of G, it is also easy to verify the following claims.

Claim 1: G is finite.

To show this, it is sufficient to show that C_{Θ_0} is finite. Consider a function $f : C_{\Theta_0} \to M$ defined by $f : \widehat{\rho} \mapsto h(\rho)$. Clearly, f is well-defined, total and injective. Therefore, because M is finite, C_{Θ_0} is also finite.

Claim 2: $G \models \Sigma_1$.

This is immediate from the construction of G.

Claim 3: $G \models \Sigma_2$.

First, by assumption, $\alpha_i \approx \beta_i$ for every $i \in [1, n]$. In addition, for every $\rho \in , {}^*_0$,

$$h(\rho \cdot \alpha_i) = h(\rho \cdot \beta_i),$$

because h is a homomorphism. Therefore, $\rho \cdot \alpha_i \approx \rho \cdot \beta_i$. That is,

$$\widehat{\rho \cdot \alpha}_i = \widehat{\rho \cdot \beta}_i.$$

Second, by the construction of G, for any $o \in |G|$, $o = o(\widehat{\rho})$ for some $\rho \in , {}^*_0$. Moreover, by the property of G described above, for each $\varrho \in , {}^*_0$, it can be shown by a straightforward induction on $|\varrho|$ that there is a unique $o' \in |G|$, such that $G \models \varrho(o(\widehat{\rho}), o')$. In addition, $o' = o(\widehat{\rho \cdot \varrho})$. Therefore, for each $o(\widehat{\rho})$ such that $G \models K(o(\widehat{\epsilon}), o(\widehat{\rho}))$, $o(\widehat{\rho \cdot \alpha}_i)$ is the unique node in $|G|$ such that

$$G \models \alpha_i(o(\widehat{\rho}), o(\widehat{\rho \cdot \alpha}_i)).$$

Similarly, we have $G \models \beta_i(o(\widehat{\rho}), o(\widehat{\rho \cdot \beta}_i))$. By $o(\widehat{\rho \cdot \alpha}_i) = o(\widehat{\rho \cdot \beta}_i)$, we have

$$G \models \beta_i(o(\widehat{\rho}), o(\widehat{\rho \cdot \alpha}_i)).$$

Therefore, for each $i \in [1, n]$, $G \models \forall x \, (K(r, x) \rightarrow \forall y \, (\alpha_i(x, y) \rightarrow \beta_i(x, y)))$. Similarly, it can be shown that $G \models \forall x \, (K(r, x) \rightarrow \forall y \, (\beta_i(x, y) \rightarrow \alpha_i(x, y)))$. Therefore, $G \models \Sigma_2$.

Claim 4: $G \not\models \forall x \, (\alpha(r, x) \rightarrow \beta(r, x))$.

As in Claim 3, we can show that

$$\begin{aligned} G &\models \alpha(o(\widehat{\epsilon}), o(\widehat{\alpha})), \\ G &\models \beta(o(\widehat{\epsilon}), o(\widehat{\beta})). \end{aligned}$$

In addition, $o(\widehat{\beta})$ is the unique node in $|G|$ such that $G \models \beta(o(\widehat{\epsilon}), o(\widehat{\beta}))$. By assumption, we have $\alpha \not\approx \beta$. Thus $\widehat{\alpha} \neq \widehat{\beta}$. Hence $o(\widehat{\alpha}) \neq o(\widehat{\beta})$. Therefore,

$$G \models \alpha(o(\widehat{\epsilon}), o(\widehat{\alpha})) \wedge \neg\beta(o(\widehat{\epsilon}), o(\widehat{\alpha})).$$

That is, $G \not\models \forall x \, (\alpha(r, x) \rightarrow \beta(r, x))$.

(*only if*) Suppose that $\Sigma_1 \cup \Sigma_2 \not\models_f \forall x\,(\alpha(r,x) \rightarrow \beta(r,x)) \ \wedge \ \forall x\,(\beta(r,x) \rightarrow \alpha(r,x))$. We show that $\Theta_0 \not\models_f (\alpha,\ \beta)$. More specifically, we define a finite monoid $(M,\ \circ,\ 1)$ and a homomorphism $h : {}^{*}_{0} \rightarrow M$ such that for every $i \in [1,n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation on ${}^{*}_{0}$. By assumption, there exists a finite σ_0-structure G, such that $G \models \Sigma_1 \cup \Sigma_2$, but

$$G \not\models \forall x\,(\alpha(r,x) \rightarrow \beta(r,x)) \ \wedge \ \forall x\,(\beta(r,x) \rightarrow \alpha(r,x)).$$

Without loss of generality, assume that there is $o \in |G|$ such that

$$G \models \alpha(r^G,\ o) \wedge \neg\beta(r^G, o).$$

Based on G, we define an equivalence relation $\sim$ on ${}^{*}_{0}$ as follows:

$$\begin{aligned} \rho \sim \varrho \quad \text{iff} \quad G \ \models \ & \forall x(K(r,x) \rightarrow \forall y\,(\rho(x,y) \rightarrow \varrho(x,y))) \wedge \\ & \forall x\,(K(r,x) \rightarrow \forall y\,(\varrho(x,y) \rightarrow \rho(x,y))). \end{aligned}$$

Then by $G \models \Sigma_2$, for every $i \in [1,n]$, $\alpha_i \sim \beta_i$. By $G \models \Sigma_1$, $G \models K(r^G,\ r^G)$. In addition, by $G \models \alpha(r^G,\ o) \wedge \neg\beta(r^G, o)$, we have $G \not\models \forall x\,(K(r,x) \rightarrow \forall y\,(\alpha(x,y) \rightarrow \beta(x,y)))$. Therefore, $\alpha \not\sim \beta$.

For every $\rho \in {}^{*}_{0}$, let $[\rho]$ denote the equivalence class of ρ with respect to $\sim$. Then clearly, for every $i \in [1,n]$, $[\alpha_i] = [\beta_i]$. However, $[\alpha] \neq [\beta]$.

Using the notion of $\sim$, we define

$$M = \{[\rho] \mid \rho \in {}^{*}_{0}\}.$$

An important property of M is described as follows.

Claim 5: M is finite.

To show this, for every $\rho \in {}^{*}_{0}$, let

$$S_\rho = \{(a,b) \mid a,b \in |G|,\ G \models K(r^G,\ a) \wedge \rho(a,\ b)\}.$$

In addition, let

$$S_G = \{S_\rho \mid \rho \in {}^{*}_{0}\}.$$

Since $S_\rho \subseteq |G| \times |G|$ and $|G|$ is finite, S_G is finite. Moreover, it is easy to verify the following:

Fact: For all $\rho, \varrho \in {}^{*}_{0}$, $\rho \sim \varrho$ iff $S_\rho = S_\varrho$.

To see that the fact holds, first assume that $\rho \sim \varrho$. Then for each $(a, b) \in S_\rho$, by the definition of S_ρ, we have

$$G \models K(r^G, a) \wedge \rho(a, b).$$

By the definition of $\sim$ and the assumption that $\rho \sim \varrho$, we have

$$G \models K(r^G, a) \wedge \varrho(a, b).$$

Hence $(a, b) \in S_\varrho$. Therefore, $S_\rho \subseteq S_\varrho$. Similarly, it can be shown that $S_\varrho \subseteq S_\rho$. Hence

$$S_\rho = S_\varrho.$$

Conversely, assume that $S_\rho = S_\varrho$. Suppose, for *reductio*, that $\rho \not\sim \varrho$. Without loss of generality, assume that $G \not\models \forall x(K(r, x) \rightarrow \forall y\,(\rho(x, y) \rightarrow \varrho(x, y)))$. Then there exist $a, b \in |G|$, such that $G \models K(r^G, a) \wedge \rho(a, b) \wedge \neg\varrho(a, b)$. That is, $(a, b) \in S_\rho$ but $(a, b) \notin S_\varrho$. Hence $S_\rho \neq S_\varrho$. This contradicts the assumption. Therefore, the fact holds.

Next, consider a function $g : M \rightarrow S_G$ defined by

$$g : [\rho] \mapsto S_\rho.$$

Using the fact above, it is easy to see that g is well-defined, total and injective. Therefore, because S_G is finite, M is also finite.

Next, we define a binary operation $\circ$ on M by

$$[\rho] \circ [\varrho] = [\rho \cdot \varrho].$$

It is easy to verify the following claims.

Claim 6: $\circ$ is well-defined.

To see this, for all $\rho_1, \rho_2, \varrho_1, \varrho_2 \in {}^{*}_{0}$ such that $\rho_1 \sim \rho_2$ and $\varrho_1 \sim \varrho_2$, we show that

$$\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2.$$

To do this, consider all $o, o_1 \in |G|$ such that

$$G \models K(r^G,\, o) \wedge \rho_1 \cdot \varrho_1(o,\, o_1).$$

Clearly, there exists $o' \in |G|$ such that

$$G \models \rho_1(o,\, o') \wedge \varrho_1(o',\, o_1).$$

By $\rho_1 \sim \rho_2$, we have $G \models \rho_2(o,\, o')$. By Lemma 9.3 and $G \models K(r^G,\, o) \wedge \rho_1(o,\, o')$, we have

$$G \models K(r^G,\, o').$$

Thus by $\varrho_1 \sim \varrho_2$, we also have $G \models \varrho_2(o',\, o_1)$. Hence $G \models \rho_2 \cdot \varrho_2(o,\, o_1)$. Therefore,

$$G \models \forall\, x\, (K(r, x) \to \forall\, y\, (\rho_1 \cdot \varrho_1(x,\, y) \to \rho_2 \cdot \varrho_2(x, y))).$$

Similarly, we can show that

$$G \models \forall\, x\, (K(r, x) \to \forall\, y\, (\rho_2 \cdot \varrho_2(x, y) \to \rho_1 \cdot \varrho_1(x,\, y))).$$

Therefore, $\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2$.

Claim 7: $\circ$ is associative.

This is because for all $[\rho], [\varrho], [\lambda] \in M$,

$$\begin{aligned}
([\rho] \circ [\varrho]) \circ [\lambda] &= [\rho \cdot \varrho] \circ [\lambda] \\
&= [\rho \cdot \varrho \cdot \lambda] \\
&= [\rho] \circ ([\varrho \cdot \lambda]) \\
&= [\rho] \circ ([\varrho] \circ [\lambda]).
\end{aligned}$$

Claim 8: $[\epsilon]$ is the identity for $\circ$. This is because for any $[\rho] \in M$,

$$[\epsilon] \circ [\rho] = [\rho] = [\rho] \circ [\epsilon].$$

These claims show that $(M,\, \circ,\, [\epsilon])$ is a finite monoid.

Finally, we define $h : , {}^{*}_{0} \rightarrow M$ by

$$h : \ \rho \mapsto [\rho].$$

Clearly, h is a homomorphism since $h(\rho \cdot \varrho) = [\rho \cdot \varrho] = [\rho] \circ [\varrho] = h(\rho) \circ h(\varrho)$.

In addition, for every $i \in [1, n]$, by $[\alpha_i] = [\beta_i]$, $h(\alpha_i) = h(\beta_i)$. Moreover, by $[\alpha] \neq [\beta]$, $h(\alpha) \neq h(\beta)$. Therefore,

$$\Theta_0 \not\models_f (\alpha, \beta).$$

This completes the proof of Lemma 9.4. ∎

From Lemma 9.4, Theorem 9.2 follows immediately.

9.2.2 The decidability of EIPs in the context of $\mathcal{M}$

In contrast to the undecidability results established above, the extended implication and finite implication problems for word constraints are decidable in cubic-time in the context of $\mathcal{M}$. The cubic-time decidability follows from Theorem 8.17, which shows that in $\mathcal{M}$, the implication and finite implication problems for P_c are decidable in cubic-time. Because for any schema Δ in $\mathcal{M}$ and any path $\alpha \in Paths(\Delta)$, $P_w(\Delta, \alpha) \subseteq P_c(\Delta)$, the cubic-time decidability of EIPs in $\mathcal{M}$ follows immediately..

9.2.3 The undecidability of EIPs in the context of $\mathcal{M}^+_f$

Next, we show that in the context of $\mathcal{M}^+_f$, EIPs remain undecidable.

Theorem 9.5: In the context of the object-oriented model $\mathcal{M}^+_f$, the extended implication and finite implication problems for word constraints are undecidable. ∎

The undecidability of the extended finite implication problem can also be established by reduction from the word problem for finite monoids. In fact, the proof of Theorem 8.1 provides such a reduction. To see this, recall the schema Δ_0 in $\mathcal{M}^+_f$ and the encoding Σ_1, Σ_2 and $\varphi_{(\alpha,\beta)}$ defined in Section 8.1. In particular,

- Σ_1 consists of the following: for each $j \in [1, m]$,

$$\begin{aligned} \forall\, x\, (a(r, x) &\rightarrow b \cdot *(r, x)) \\ \forall\, x\, (b \cdot * \cdot l_j(r, x) &\rightarrow b \cdot *(r, x)) \end{aligned}$$

- Σ_2 consists of the following: for each $j \in [1, m]$,

$$\forall\, x\, (b \cdot *(r, x) \rightarrow \forall\, y\, (\alpha_i(x, y) \rightarrow \beta_i(x, y)))$$

- $\varphi_{(\alpha,\beta)} = \forall\, x(a \cdot \alpha(r, x) \rightarrow a \cdot \beta(r, x))$.

Obviously, $\Sigma_1 \cup \{\varphi_{(\alpha,\beta)}\} \subseteq P_w(\Delta_0)$ and $\Sigma_2 \subseteq P_w^{b\cdot *}(\Delta_0)$. Hence

$$\Sigma_1 \cup \Sigma_2 \cup \{\varphi_{(\alpha,\beta)}\} \subseteq P_w(\Delta_0,\, b \cdot *).$$

Therefore, by Lemma 8.7, this encoding is a reduction from the word problem for finite monoids to the extended finite implication problem for word constraints.

As a result, the undecidability of the extended implication problem for word constraints follows from Lemma 8.8. Indeed, Lemma 8.8 shows that for any schema Δ in $\mathcal{M}_f^+$ and every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$, $\Sigma \models_\Delta \varphi$ iff $\Sigma \models_{(f,\,\Delta)} \varphi$. Therefore, the extended implication problem and the extended finite implication problem for word constraints coincide in the context of $\mathcal{M}_f^+$. Thus the extended implication problem for word constraints is also undecidable.

9.2.4 The undecidability of EIPs in the context of $\mathcal{M}^+$

In the context of $\mathcal{M}^+$, EIPs are also undecidable.

Theorem 9.6: In the context of the object-oriented model $\mathcal{M}^+$, the extended implication and finite implication problems for word constraints are undecidable. ∎

The proof of this theorem is the same as the argument for Theorem 8.13. Recall that Theorem 8.13 is established by reduction from the word problem for monoids and the word problem for finite monoids. The encoding in the reduction involves constraints in $P_w(\Delta_0,\, b \cdot *)$ only. Therefore, from Lemma 8.15, Theorem 9.6 follows immediately.

9.3 Decidability and undecidability of PBIPs

In this section, we study PBIPs (the prefix bounded implication and finite implication problems for path constraints). The main results of this section are the following.

1. In the context of the semistructured data model $\mathcal{SM}$, PBIPs are decidable in PTIME.
2. In contrast, the same problems become undecidable in the context of the object-oriented model $\mathcal{M}^+$.
3. In the context of the object-oriented model $\mathcal{M}_f^+$, these problems are also undecidable.
4. In the context of the object-oriented model $\mathcal{M}$, PBIPs are decidable in cubic-time.

9.3.1 The decidability of PBIPs in the context of $\mathcal{SM}$

We first establish the decidability of the prefix bounded implication and finite implication problems for P_c in the context of $\mathcal{SM}$.

Theorem 9.7: In the context of the semistructured data model $\mathcal{SM}$, the prefix bounded implication and finite implication problems for path constraints are decidable in PTIME. ∎

To simplify the proof, we consider a special case of the prefix bounded (finite) implication problem, which is defined below.

Definition 9.10: Let K be a binary relation symbol. A *constraint restricted by* K is a constraint of P_c bounded by the empty path ϵ and K.

A *subset of* P_c *restricted by* K is a finite subset of P_c with prefix bounded by ϵ and K. ∎

Definition 9.11: In the context of $\mathcal{SM}$, the *simple prefix bounded (finite) implication problem for path constraints* (P_c) is the problem of determining, given a subset $\Sigma \cup \{\varphi\}$ of P_c restricted by a binary relation symbol K, where φ is also restricted by K, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$). ∎

The lemmas below allow us to consider the simple prefix bounded (finite) implication

problem for P_c only.

Lemma 9.8: In the context of $\mathcal{SM}$, the prefix bounded (finite) implication problem for P_c is decidable in PTIME if the simple prefix bounded (finite) implication problem for P_c is decidable in PTIME. ∎

Lemma 9.9: In the context of $\mathcal{SM}$, the simple prefix bounded implication and finite implication problems for path constraints are decidable in PTIME. ∎

To prove Lemma 9.8, we first generalize the function δ given in Section 9.1 as follows.

Let α be a path and φ be a constraint in P_c. Then we define $\delta(\varphi, \alpha)$ to be

- $\forall x\,(\alpha \cdot pf(\varphi)(r, x) \rightarrow \forall y\,(lt(\varphi)(x, y) \rightarrow rt(\varphi)(x, y)))$, if φ is a forward constraint, or
- $\forall x\,(\alpha \cdot pf(\varphi)(r, x) \rightarrow \forall y\,(lt(\varphi)(x, y) \rightarrow rt(\varphi)(y, x)))$, if φ is a backward constraint.

For every subset Σ of P_c, the set $\{\delta(\varphi, \alpha) \mid \varphi \in \Sigma\}$ is called the *extension of* Σ *with common prefix* α.

Lemma 9.8 follows from the lemma below.

Lemma 9.10: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c, and α be a path. Then in the context of $\mathcal{SM}$,

$$\Sigma \models \varphi \quad \text{iff} \quad \{\delta(\phi, \alpha) \mid \phi \in \Sigma\} \models \delta(\varphi, \alpha), \tag{a}$$

$$\Sigma \models_f \varphi \quad \text{iff} \quad \{\delta(\phi, \alpha) \mid \phi \in \Sigma\} \models_f \delta(\varphi, \alpha). \tag{b}$$

∎

Using Lemma 9.10, we prove Lemma 9.8 as follows. Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c with prefix bounded by a path α and a binary relation symbol K. Then by Definition 9.5, $\Sigma \cup \{\varphi\}$ is in fact the extension of a set $\Sigma' \cup \{\varphi'\}$ of P_c with common prefix α, where $\Sigma' \cup \{\varphi'\}$ is a subset of P_c restricted by K. More specifically, $\Sigma = \{\delta(\phi, \alpha) \mid \phi \in \Sigma'\}$ and $\varphi = \delta(\varphi', \alpha)$. In addition, given $\Sigma \cup \{\varphi\}$, α can be determined in linear-time, and given α, $\Sigma' \cup \{\varphi'\}$ can be computed in linear-time. Thus by Lemma 9.10, if the simple prefix bounded (finite) implication problem for P_c is decidable in PTIME, then so is the prefix bounded (finite) implication problem for P_c.

To prove Lemma 9.9, recall the following from [9].

Lemma 9.11 [9]: In the context of $\mathcal{SM}$, the implication and finite implication problems for word constraints are decidable in PTIME. ∎

In addition, we also need the following lemma. Note that by Definition 9.5, a subset Σ of P_c restricted by K can be partitioned into Σ_K and Σ_r, where

$$\begin{aligned} \Sigma_K &= \{\varphi \mid \varphi \in \Sigma,\ \varphi \text{ is restricted by } K\}, \\ \Sigma_r &= \Sigma \setminus \Sigma_K. \end{aligned}$$

We call Σ_K the *critical subset of* Σ *with respect to* K.

Lemma 9.12: Let $\Sigma \cup \{\varphi\}$ be subset of P_c restricted by K, Σ_K be the critical subset of Σ with respect to K, and $\varphi \in \Sigma_K$. Let $\Sigma'_K = \Sigma_K \setminus \{\varphi\}$. Then in the context of $\mathcal{SM}$,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma'_K \models \varphi \tag{a}$$

$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma'_K \models_f \varphi. \tag{b}$$

∎

Next, we prove Lemma 9.9 by reducing the simple bounded (finite) implication problem for P_c to the (finite) implication problem for P_w. To do this, we define a mapping $\omega : P_c \to P_w$ as follows: For each $\phi \in P_c$,

$$\omega : \phi \mapsto \forall\, x\, (lt(\phi)(r, x) \to rt(\phi)(r, x)).$$

Here the notations pf, lt and rt are described in Definition 2.2. It should be noted that ω is computable in time linear in the length of ϕ. Let $\Sigma \cup \{\varphi\}$ be a subset of P_c restricted by K, which is partitioned into Σ_K and Σ_r as described above, and $\varphi \in \Sigma_K$. Let $\Sigma'_K = \Sigma_K \setminus \{\varphi\}$. By Definition 9.5, all the constraints in Σ_K have the common prefix K. That is, each constraint $\phi \in \Sigma_K$ has the form:

$$\forall\, x\, (K(r, x) \to \forall\, y\, (\rho(x, y) \to \varrho(x, y))).$$

Thus we have $\phi = \delta(\omega(\phi),\ K)$. Hence by Lemmas 9.10 and 9.12, the reduction can be given as:

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma'_K \models \varphi \quad \text{iff} \quad \{\omega(\phi) \mid \phi \in \Sigma'_K\} \models \omega(\varphi)$$

$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma'_K \models_f \varphi \quad \text{iff} \quad \{\omega(\phi) \mid \phi \in \Sigma'_K\} \models_f \omega(\varphi)$$

Therefore, the PTIME decidability of the simple prefix bounded (finite) implication problem for P_c follows from Lemma 9.11.

Finally, we prove Lemmas 9.10 and 9.12.

Proof of Lemma 9.10: The proof below is similar to the proof of Proposition 5.10. We show (b) only. The proof of (a) is similar and simpler.

To prove (b), it is sufficient to show that

$$\bigwedge \Sigma \wedge \neg\varphi \text{ has a finite model} \quad \text{iff} \quad \bigwedge_{\phi \in \Sigma} \delta(\phi,\, \alpha) \wedge \neg\delta(\varphi,\, \alpha) \text{ has a finite model.}$$

To simplify the discussion, assume that all the constraints in $\Sigma \cup \{\varphi\}$ are of the forward form. The proof for the backward case is similar.

(if) Suppose that $\bigwedge_{\phi \in \Sigma} \delta(\phi,\, \alpha) \wedge \neg\delta(\varphi,\, \alpha)$ has a finite model $G = (|G|,\, r^G,\, E^G)$. Then we construct a finite model of $\bigwedge \Sigma \wedge \neg\varphi$. That is, we construct a finite σ-structure H such that $H \models \bigwedge \Sigma \wedge \neg\varphi$. Here σ is the signature defined in Section 2.1.

Since $G \models \neg\delta(\varphi,\, \alpha)$, i.e.,

$$G \models \exists\, x\, y\, (\alpha \cdot pf(\varphi)(r^G, x) \wedge lt(\varphi)(x, y) \wedge \neg rt(\varphi)(x, y)),$$

there exist $a, b, c \in |G|$, such that

$$G \models \alpha(r^G, a) \wedge pf(\varphi)(a, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

Let $m = max\{|pf(\phi)| + |lt(\phi)|,\ |pf(\phi)| + |rt(\phi)| \mid \phi \in \Sigma \cup \{\varphi\}\} \ + \ 1$. Then using a, m and G, we define $H = (|H|,\, r^H,\, E^H)$ as follows.

- $|H| = \{o \mid o \in |G|,\, G \models \rho(a,\, o),\, \rho \text{ is a path and } |\rho| \leq m\}$.
- $r^H = a$.

- For every $K \in E$ and all $o, o' \in |H|$, $H \models K(o,\, o')$ iff $G \models K(o,\, o')$. Here E is the set of binary relation symbols in σ.

It is easy to verify the following.

(1) H is finite. This is because $|G|$ is finite and $|H| \subseteq |G|$.

(2) $H \models \neg\varphi$.

Since $|pf(\varphi)| + |lt(\varphi)| < m$, we have that $b \in |H|$ and $c \in |H|$. Thus by the definition of H, we have

$$H \models pf(\varphi)(a, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

That is, $H \models \neg\varphi$.

(3) $H \models \Sigma$.

We show this by *reductio*. Suppose that there exists $\phi \in \Sigma$, such that $H \models \neg\phi$. Then there exist $d, e \in |H|$, such that

$$H \models pf(\phi)(a, d) \wedge lt(\phi)(d, e) \wedge \neg rt(\phi)(d, e).$$

Since $|pf(\phi)| + |lt(\phi)| < m$ and $|pf(\phi)| + |rt(\phi)| < m$, by the definition of H, we have that

$$G \models \alpha(r^G, a) \wedge pf(\phi)(a, d) \wedge lt(\phi)(d, e) \wedge \neg rt(\phi)(d, e).$$

That is, $G \models \neg\delta(\phi, \alpha)$. This contradicts the assumption that $G \models \{\delta(\phi, \alpha) \mid \phi \in \Sigma\}$.

(*only if*) Suppose that $\bigwedge \Sigma \wedge \neg\varphi$ has a finite model $G = (|G|,\, r^G,\, E^G)$. Then we show that $\bigwedge_{\phi \in \Sigma} \delta(\phi,\, \alpha) \wedge \neg\delta(\varphi,\, \alpha)$ has a finite model $H = (|H|,\, r^H,\, E^H)$, as follows. Let

$$L_\alpha = \{\rho \mid \rho \text{ is a path, } \rho \prec_p \alpha\}.$$

Here $\rho \prec_p \alpha$ denotes that ρ is a proper prefix of α, as described in Chapter 2. For each $\rho \in L_\alpha$, let c_ρ be a distinct node not in $|G|$. Let

- $|H| = |G| \cup \{c_\rho \mid \rho \in L_\alpha\}$;

- $r^H = c_\epsilon$;
- For all $a, b \in |H|$ and each $K \in E$, $H \models K(a, b)$ iff one of the following conditions is satisfied:
  - there exists $\rho \in L_\alpha$, such that $a = c_\rho$ and $b = c_{\rho \cdot K}$ and $\rho \cdot K \in L_\alpha$; or
  - there exists $\rho \in L_\alpha$, such that $\alpha = \rho \cdot K$ and $a = c_\rho$ and $b = r^G$; or
  - $a, b \in |G|$ and $G \models K(a, b)$.

It is easy to verify the following.

(1) H is finite. This is because both $|G|$ and L_α are finite.

(2) $H \models \neg\delta(\varphi, \alpha)$.

To show this, we first observe the following simple facts, which are immediate from the construction of H.

Fact 1: r^G is the unique node in $|H|$ such that $H \models \alpha(c_\epsilon, r^G)$. In addition, for all $\rho, \varrho \in L_\alpha$ and $K \in E$,

$$H \models K(c_\rho, c_\varrho) \quad \text{iff} \quad \varrho = \rho \cdot K.$$

Fact 2: For each $\rho \in L_\alpha$, $K \in E$ and each $a \in |G| \setminus \{r^G\}$, we have

$$H \models \neg K(c_\rho, a) \wedge \neg K(a, c_\rho).$$

In addition, if $\alpha \neq \rho \cdot K$, then

$$H \models \neg K(c_\rho, r^G) \wedge \neg K(r^G, c_\rho).$$

However, if $\alpha = \rho \cdot K$, then

$$H \models K(c_\rho, r^G) \wedge \neg K(r^G, c_\rho).$$

Using these facts, we show that $H \models \neg\delta(\varphi, \alpha)$. By $G \models \neg\varphi$, there exist $b, c \in |G|$, such that

$$G \models pf(\varphi)(r^G, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

By Facts 1 and 2,

$$H \models \alpha(c_\epsilon,\, r^G) \wedge pf(\varphi)(r^G,\, b) \wedge lt(\varphi)(b,\, c) \wedge \neg rt(\varphi)(b,\, c).$$

That is, $H \models \neg\delta(\varphi,\, \alpha)$.

(3) $H \models \{\delta(\phi,\, \alpha) \mid \phi \in \Sigma\}$.

Suppose for *reductio* that there exists $\phi \in \Sigma$ such that $H \models \neg\delta(\phi,\, \alpha)$. Then there exist $a, b, c \in |H|$, such that

$$H \models \alpha(c_\epsilon,\, a) \wedge pf(\phi)(a,\, b) \wedge lt(\phi)(b,\, c) \wedge \neg rt(\phi)(b,\, c).$$

By Fact 1, $a = r^G$. By Fact 2, $b, c \in |G|$, and moreover, by the construction of H, we have

$$G \models pf(\phi)(a,\, b) \wedge lt(\phi)(b,\, c) \wedge \neg rt(\phi)(b,\, c).$$

That is, $G \models \neg\phi$. This contradicts the assumption that $G \models \Sigma$.

This completes the proof of Lemma 9.10. ∎

Proof of Lemma 9.12: We show (b) only. The proof of (a) is similar and simpler.

The *if* direction can be verified by using the argument for Lemma 9.10. For the *only if* direction, suppose that $\bigwedge \Sigma'_K \wedge \neg\varphi$ has a finite model. Then we show that $\Sigma \wedge \neg\varphi$ also has a finite model

$$H = (|H|,\, r^H,\, E^H).$$

By Definitions 9.5 and 9.10, for every $\phi \in \Sigma_K$, $pf(\phi) = K$. In addition, $lt(\phi) \neq \epsilon$ and K is not a prefix of $lt(\phi)$. Recall the function $\omega : P_c \to P_w$ defined above. It is easy to see that Σ_K is the extension of the set $\{\omega(\phi) \mid \phi \in \Sigma_K\}$ with common prefix K. By Lemma 9.10 and the assumption that $\bigwedge \Sigma'_K \wedge \neg\varphi$ has a finite model, we have that $\bigwedge_{\phi \in \Sigma'_K} \omega(\phi) \wedge \neg\omega(\varphi)$ also has a finite model. In addition, by the proof of lemma 9.10, if $\bigwedge_{\phi \in \Sigma'_K} \omega(\phi) \wedge \neg\omega(\varphi)$ has a finite model $G_\omega = (|G_\omega|,\, r^{G_\omega},\, E^{G_\omega})$, then $\bigwedge \Sigma'_K \wedge \neg\varphi$ has a finite model

$$G = (|G|,\, r^G,\, E^G),$$

such that G has the following properties:

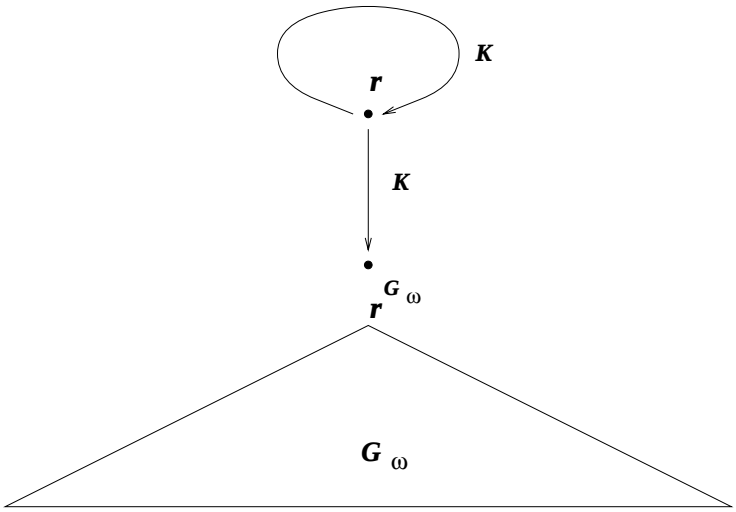

Figure 9.2: The structure H in the proof of Lemma 9.12

- $|G| = |G_\omega| \cup \{r^G\}$ and $r^G \notin |G_\omega|$.
- E^G is E^{G_ω} augmented by the following: $G \models K(r^G, r^{G_\omega})$. In other words, E^G is $E^{G_\omega} \cup \{K(r^G, r^{G_\omega})\}$.

Using G, we can define a finite model H of $\Sigma \wedge \neg\varphi$ such that $|H| = |G|$, $r^H = r^G$, and $E^H = E^G \cup \{K(r^H, r^H)\}$. The structure H is shown in Figure 9.2.

Clearly, H and G have the same size. Therefore, H is finite.

Below we show that H is a model of $\Sigma \wedge \neg\varphi$. Let $\Sigma_r = \Sigma \setminus \Sigma_K$.

(1) $H \models \Sigma_r$.

For every $\phi \in \Sigma_r$, by Definitions 9.5 and 9.10, K is not a prefix of $pf(\phi)$. Therefore, if $pf(\phi) \neq \epsilon$, then by the construction of H, there are no $o, o' \in |H|$, such that

$$H \models pf(\phi)(r^H, o) \wedge lt(\phi)(o, o').$$

Therefore, $H \models \phi$. If $pf(\phi) = \epsilon$, then by Definition 9.5, ϕ is

$$\forall x\, (\epsilon(r, x) \rightarrow K(r, x)).$$

Clearly, $H \models \phi$ since $(r^H, r^H) \in K^H$. Therefore, $H \models \Sigma_r$.

(2) $H \models \Sigma'_K$.

Suppose, for *reductio*, that there is $\phi \in \Sigma'_K$ such that $H \models \neg\phi$. Since ϕ is restricted by K, by Definitions 9.5 and 9.10, ϕ is of the form

$$\forall x\,(K(r,\, x) \rightarrow \forall y\,(\beta(x,\, y) \rightarrow \gamma(x,\, y))),$$

where $K \not\preceq_p \beta$ and $\beta \neq \epsilon$. Thus by the definition of H, if $H \models \neg\phi$, then there exists $o \in |G_\omega|$, such that

$$H \models K(r^H,\, r^{G_\omega}) \wedge \beta(r^{G_\omega},\, o) \wedge \neg\gamma(r^{G_\omega},\, o).$$

In addition, we have

$$G_\omega \models \beta(r^{G_\omega},\, o) \wedge \neg\gamma(r^{G_\omega},\, o).$$

That is, $G_\omega \models \neg\omega(\phi)$. This contracts the assumption that structure G_ω is a model of $\bigwedge_{\phi\in\Sigma'_K} \omega(\phi) \wedge \neg\omega(\varphi)$. Therefore, $H \models \phi$. Hence $H \models \Sigma'_K$.

(3) $H \models \neg\varphi$.

Note that φ is restricted by K. Hence φ is of the form

$$\forall x\,(K(r,\, x) \rightarrow \forall y\,(\beta(x,\, y) \rightarrow \gamma(x,\, y))),$$

where $K \not\preceq_p \beta$ and $\beta \neq \epsilon$. By $G \models \neg\varphi$, there are $o, o' \in |G|$, such that

$$G \models K(r^G,\, o) \wedge \beta(o,\, o') \wedge \neg\gamma(o,\, o').$$

Clearly, by the definition of G, $o = r^{G_\omega}$ and $o' \in |G_\omega|$. By the definition of H, $r^H = r^G$ and in addition,

$$H \models K(r^H,\, r^{G_\omega}) \wedge \beta(r^{G_\omega},\, o') \wedge \neg\gamma(r^{G_\omega},\, o').$$

That is, $H \models \neg\varphi$.

This completes the proof of Lemma 9.12. ∎

The following should be noted.

(1) The proof above is not applicable in the context of $\mathcal{M}$, $\mathcal{M}^+$ or $\mathcal{M}^+_f$. More specifically, for any schema Δ in these models, the structure H shown in Figure 9.2 is not in $\mathcal{U}(\Delta)$, because of the type constraint $\Phi(\Delta)$. To see this, suppose, for *reductio*, that $H \in \mathcal{U}(\Delta)$ for some $\Delta = (\mathcal{C}, \nu, DBtype)$. Since $H \models \Phi(\Delta)$ and r^H is the root node of H, we must have

$$r^H \in R^H_{DBtype}.$$

In $\mathcal{M}^+_f$, $\mathcal{M}^+$ and $\mathcal{M}$, $DBtype$ must be either a record or a set type, but cannot be a class type. In addition, a record or a set type cannot be defined in terms of itself. More specifically, if $DBtype = \{\tau\}$, then $\tau \neq DBtype$, and if $DBtype = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then for every $i \in [1, n]$, $\tau_i \neq DBtype$. Moreover, every node in H has a unique type. Because of this, if $H \models K(r^H, o)$, then $o \notin R^H_{DBtype}$. However, $H \models K(r^H, r^H)$. Hence $r^H \notin R^H_{DBtype}$. This leads to a contradiction.

It should be mentioned that for a structure $G \in \mathcal{U}(\Delta)$ and $o \in |G|$, if $o \in R^G_C$ for some class $C \in \mathcal{C}$, then it is possible that $G \models K(o, o)$. This may occur when, for example, $\nu(C) = [K : C, \ldots]$. However, in this case, o has a unique outgoing edge labeled K. In other words, if $G \models K(o, o)$ then there is no $o' \in |G|$ such that $o' \neq o$ and $G \models K(o', o)$. Note that $H \models K(r^H, r^H) \wedge K(r^H, r^{G_\omega})$ and $r^H \neq r^{G_\omega}$. Hence even if $r^H \in R^G_C$, H still cannot be in $\mathcal{U}(\Delta)$ as long as $\nu(C)$ is a record type.

(2) The proof of Theorem 9.7 does not conflict with Theorem 9.1. Recall Σ_1, Σ_2, $\varphi_{(\alpha,\beta)}$ and $\varphi_{(\beta,\alpha)}$ defined in Section 9.2. The set $\Sigma_1 \cup \Sigma_2 \cup \{\varphi_{(\alpha,\beta)}, \varphi_{(\beta,\alpha)}\}$ is not a prefix bounded subset of P_c.

9.3.2 The undecidability of PBIPs in the context of $\mathcal{M}^+$

In contrast to the decidability results established above, below we show that in the context of the object-oriented model $\mathcal{M}^+$, PBIPs are undecidable.

Theorem 9.13: In the context of $\mathcal{M}^+$, the prefix bounded implication and finite implication problems for path constraints are undecidable. ∎

The proof is similar to that of Theorem 9.6, by reduction from the word problem for (finite) monoids. Recall the alphabet $, _0$ described in Section 8.1. Using $, _0$, we define a schema Δ_1 in $\mathcal{M}^+$ as follows:

$$\Delta_1 = (\mathcal{C},\ \nu,\ DBtype),$$

where

- $\mathcal{C} = \{C, C_s, C_l\}$,
- ν is defined by:

$$\begin{aligned} C &\mapsto [l_1 : C,\ \ldots,\ l_m : C] \\ C_s &\mapsto \{C\} \\ C_l &\mapsto [a : C,\ b : C_s,\ K : C_l] \end{aligned}$$

  where $a, b, K \notin , _0$.

- $DBtype = [l : C_l]$, where $l \notin , _0$.

We encode equations over $, _0$ in terms of constraints in $P_c(\Delta_1)$. Recall the finite set Θ_0 of equations described in Section 8.1. We encode Θ_0 in terms of a finite set Σ, which consists of the following constraints of $P_c(\Delta_1)$:

1. $\forall x\,(l(r,x) \rightarrow \forall y\,(\epsilon(x,y) \rightarrow K(x,y)))$;
2. $\forall x\,(l \cdot K(r,x) \rightarrow \forall y\,(a(x,y) \rightarrow b \cdot *(x,y)))$;
3. for each $j \in [1,m]$, $\forall x\,(l \cdot K(r,x) \rightarrow \forall y\,(b \cdot * \cdot l_j(x,y) \rightarrow b \cdot *(x,y)))$;
4. for each $(\alpha_i,\ \beta_i) \in \Theta_0$, $\forall x\,(l \cdot b \cdot *(r,x) \rightarrow \forall y\,(\alpha_i(x,y) \rightarrow \beta_i(x,y)))$.

We encode a test equation $(\alpha,\ \beta)$ over $, _0$ with the constraint

$$\varphi_{(\alpha,\beta)} = \forall x\,(l \cdot K(r,x) \rightarrow \forall y\,(a \cdot \alpha(x,y) \rightarrow a \cdot \beta(x,y))).$$

It is easy to see that $\Sigma \cup \{\varphi_{(\alpha,\beta)}\}$ is a subset of $P_c(\Delta_1)$ with prefix bounded by l and K. More specifically, this set can be partitioned into Σ_r and Σ_K such that

- Σ_r consists of the constraints specified in (1) and (4), and
- Σ_K consists of $\varphi_{(\alpha,\beta)}$ as well as those defined in (2) and (3).

All the constraints in Σ_K are bounded by l and K, while the constraints in Σ_r are not. In addition, for any $\phi \in \Sigma_r$, $pf(\phi)$ is either $l \cdot b \cdot *$ or l. In particular, if $pf(\phi) = l$, then ϕ is the constraint given in (1).

We reduce the word problem for monoids to the problem of determining whether

$$\Sigma \models_{\Delta_1} \varphi_{(\alpha,\beta)},$$

and analogously, reduce the word problem for finite monoids to the problem of determining whether

$$\Sigma \models_{(f,\,\Delta_1)} \varphi_{(\alpha,\beta)}.$$

As in Section 9.2, it is easy to verify the following lemmas.

Lemma 9.14: For each $G \in \mathcal{U}(\Delta_1)$, G has the following properties.

1. There is a unique node $o \in |G|$, such that $G \models l(r^G, o)$. This node is denoted by o_l.
2. There exists a unique node $o_K \in |G|$, such that $G \models l \cdot K(r^G, o_K)$. In addition, if $G \models \forall x\,(l(r,x) \rightarrow \forall y\,(\epsilon(x,y) \rightarrow K(x,y)))$, then $o_K = o_l$.
3. For every $\alpha \in {}^{*}_{0}$, the following statements hold.
   (a) There is a unique $o \in |G|$ such that $G \models a \cdot \alpha(o_l, o)$. This node is denoted by o_α.
   (b) For every $o \in |G|$, if $G \models R^G_C(o)$, then there is a unique $o' \in |G|$, such that $G \models \alpha(o, o')$.

■

Lemma 9.15: For every $G \in \mathcal{U}(\Delta_1)$ and all $\alpha, \beta \in {}^{*}_{0}$, if

$$G \models \forall x\,(l \cdot b \cdot *(r,x) \rightarrow \forall y\,(\alpha(x,y) \rightarrow \beta(x,y))),$$

then

$$G \models \forall\, x\, (l \cdot b \cdot *(r, x) \rightarrow \forall\, y\, (\beta(x, y) \rightarrow \alpha(x, y))).$$

Similarly, if

$$G \models \forall\, x\, (l \cdot K(r, x) \rightarrow \forall\, y\, (a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y))),$$

then

$$G \models \forall\, x\, (l \cdot K(r, x) \rightarrow \forall\, y\, (a \cdot \beta(x, y) \rightarrow a \cdot \alpha(x, y))).$$

■

Lemma 9.16: For every $G \in \mathcal{U}(\Delta_1)$, if $G \models \Sigma$, then for every $\alpha \in {}^{*}_{0}$, we have

$$G \models l \cdot b \cdot *(r^G,\, o_\alpha),$$

where o_α is the unique node in $|G|$ such that $G \models l \cdot a \cdot \alpha(r^G,\, o_\alpha)$, as specified in Lemma 9.14. In addition, for every $o, o' \in |G|$ such that $G \models l \cdot b \cdot *(r^G,\, o') \wedge \alpha(o',\, o)$, we have

$$G \models l \cdot b \cdot *(r^G,\, o).$$

■

Lemma 9.17: For every $G \in \mathcal{U}(\Delta_1)$, if $G \models \Sigma$ and for some $\alpha, \beta \in {}^{*}_{0}$,

$$G \models \forall\, x\, (l \cdot b \cdot *(r, x) \rightarrow \forall\, y\, (\alpha(x, y) \rightarrow \beta(x, y))),$$

then

$$G \models \forall\, x\, (l \cdot K(r, x) \rightarrow \forall\, y\, (a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y))).$$

■

Next, we show that the encoding given above is indeed a reduction from the word problem for (finite) monoids.

Lemma 9.18: In the context of $\mathcal{M}^+$, for all α and β in ${}^{*}_{0}$,

$$\Theta_0 \models (\alpha,\, \beta) \;\text{ iff }\; \Sigma \models_{\Delta_1} \forall\, x\, (l \cdot K(r, x) \rightarrow \forall\, y\, (a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y))), \qquad \text{(a)}$$

$$\Theta_0 \models_f (\alpha,\, \beta) \;\text{ iff }\; \Sigma \models_{(f,\, \Delta_1)} \forall\, x\, (l \cdot K(r, x) \rightarrow \forall\, y\, (a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y))). \qquad \text{(b)}$$

■

Proof: The proof is similar to that of Lemma 8.7. We prove (b) only. The proof of (a) is similar and simpler.

(*if*) Suppose that $\Theta_0 \not\models_f (\alpha, \beta)$. Then we construct a $\sigma(\Delta_1)$-structure $G \in \mathcal{U}_f(\Delta_1)$, such that $G \models \Sigma$ and $G \not\models \forall x\,(l \cdot K(r, x) \rightarrow \forall y\,(a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y)))$.

By $\Theta_0 \not\models_f (\alpha, \beta)$, there exist a finite monoid $(M, \circ, 1)$ and a homomorphism $h : {}^{*}_{0} \rightarrow M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Using M and h, we define an equivalence relation $\approx$ in the same way as in the proof of Lemma 8.7. In addition, for each $\rho \in {}^{*}_{0}$, let $\widehat{\rho}$ be defined as in the proof of Lemma 8.7. Similarly, we define C_{Θ_0}.

Using C_{Θ_0}, we define $G = (|G|, r^G, E^G, R^G)$ as follows.

(1) $|G|$.

For each $\rho \in {}^{*}_{0}$, let $o(\widehat{\rho})$, o_r and o_b be defined as in the proof of Lemma 8.7. In addition, let o_l be a distinct node. Then we define

$$|G| = \{o_r, o_b, o_l\} \cup \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}.$$

(2) $r^G = o_r$.

(3) The unary relations R^G_C, $R^G_{C_s}$ and R^G_{DBtype} are defined in the same way as in the proof of Lemma 8.7. In addition, let $R^G_{C_l} = \{o_l\}$.

(4) The binary relations are populated as follows.

- $G \models l(o_r, o_l)$.
- $G \models K(o_l, o_l)$.
- $G \models a(o_l, o(\widehat{\epsilon}))$.
- $G \models b(o_l, o_b)$.
- For each $\widehat{\rho} \in C_{\Theta_0}$, let $G \models *(o_b, o(\widehat{\rho}))$.

  In addition, for each $j \in [1, m]$, let $G \models l_j(o(\widehat{\rho}), o(\widehat{\rho \cdot l_j}))$.

The structure G is shown in Figure 9.3. It should be noted that since $o_l \in R^G_{C_l}$, it is valid to have $G \models K(o_l, o_l)$. However, since $\nu(C_l)$ is a record type, there is no $o \in |G|$ such that $o \neq o_l$ and $G \models K(o_l, o_l) \wedge K(o_l, o)$.

As in proof of Lemma 8.7, it is easy to verify that

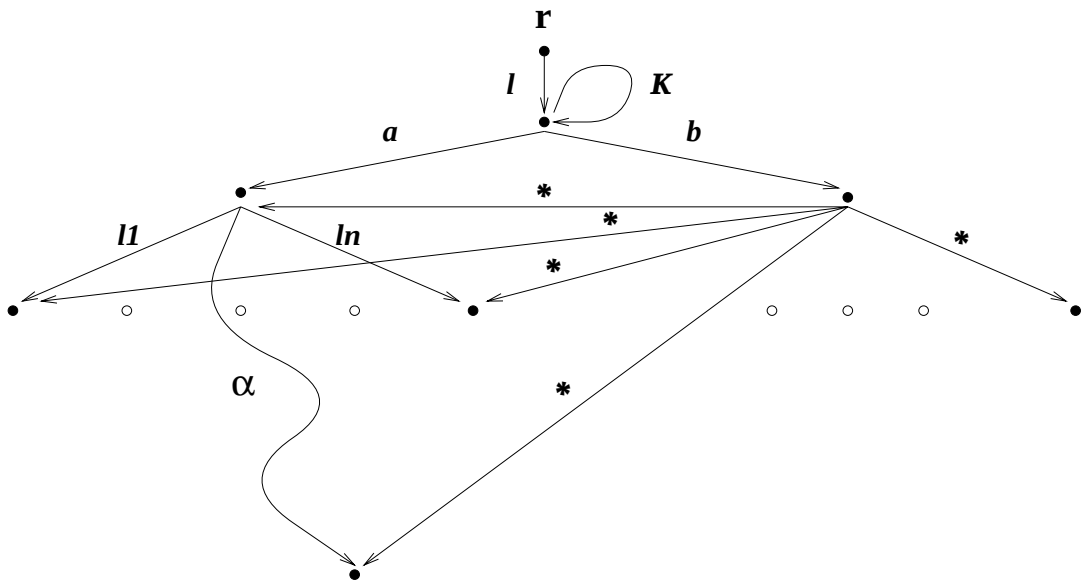

Figure 9.3: The structure G in the proof of Lemma 9.18

- $G \in \mathcal{U}_f(\Delta_1)$,
- $G \models \Sigma$, and
- $G \not\models \forall x\,(l \cdot K(r,x) \rightarrow \forall y\,(a \cdot \alpha(x,y) \rightarrow a \cdot \beta(x,y)))$.

(*only if*) Suppose that there is $G \in \mathcal{U}_f(\Delta_1)$, such that $G \models \Sigma$, but

$$G \not\models \forall x\,(l \cdot K(r,x) \rightarrow \forall y\,(a \cdot \alpha(x,y) \rightarrow a \cdot \beta(x,y))).$$

Then we show that $\Theta_0 \not\models_f (\alpha, \beta)$. That is, we show that there exist a finite monoid $(M, \circ, 1)$ and a homomorphism $h : {}^{*}_{0} \rightarrow M$ such that for any $i \in [1,n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation on ${}^{*}_{0}$ as follows:

$$\rho \sim \varrho \quad \text{iff} \quad G \models \forall x\,(l \cdot b \cdot *(r,x) \rightarrow \forall y\,(\rho(x,y) \rightarrow \varrho(x,y))).$$

As in the proof of Lemma 8.7, it can be shown that for every $i \in [1,n]$, $\alpha_i \sim \beta_i$. In addition, $\alpha \not\sim \beta$.

As in the proof of Lemma 8.7, we define $[\rho]$ for every $\rho \in {}^{*}_{0}$, and using the notion of $[\rho]$, define M and $\circ$. In addition, as in the proof of Lemma 8.7, the following can be verified.

- M is finite.
- $\circ$ is well-defined.
- $\circ$ is associative.
- $[\epsilon]$ is the identity for $\circ$.

Therefore, $(M, \circ, [\epsilon])$ is a finite monoid.

We define $h : {}^{*}_{0} \rightarrow M$ by

$$h : \rho \mapsto [\rho].$$

As in the proof of Lemma 8.7, the following can be verified.

- h is a homomorphism.
- For every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$.
- $h(\alpha) \neq h(\beta)$.

Therefore, $\Theta_0 \not\models_f (\alpha, \beta)$.

This completes the proof of Lemma 9.18. ∎

9.3.3 The undecidability of PBIPs in the context of $\mathcal{M}_f^+$

In the context of $\mathcal{M}_f^+$, PBIPs are also undecidable.

Theorem 9.19: In the context of the object-oriented model $\mathcal{M}_f^+$, the prefix bounded implication and finite implication problems for path constraints are undecidable. ∎

The proof of the undecidability of the prefix bounded finite implication problem for P_c is the same as that of Theorem 9.13. By Lemma 8.8, the undecidability of the prefix bounded implication problem for P_c follows from the undecidability of the prefix bounded finite implication problem for P_c.

9.3.4 The decidability of PBIPs in the context of $\mathcal{M}$

Obviously, the prefix bounded implication and finite implication problems for P_c are special cases of the implication and finite implication problems for P_c. As an immediate result of Theorem 8.17, PBIPs are decidable in cubic-time.

Chapter 10

Equality, Type and Word Constraints

In the previous chapters, path constraint implication has been investigated in the context of data models supporting *identity equality*. That is, in these models it is assumed that each data entity has a unique identity, and two data entities are equal if and only if they have the same identify. However, this is not the case in some object-oriented database systems. In these systems, complex values with nested structures are common. Such a complex value may not have a unique identity. The equality relation on these values is defined to be *complex value equality*, rather than identity equality. A natural question here is whether complex value equality interacts with path constraints.

This chapter studies the impact of complex value equality on path constraint implication. Complex value equality and its interaction with path constraints are addressed in Section 10.1. In Section 10.2, an object-oriented data model supporting complex values with nested structures, $\mathcal{M}^{\approx}$, is introduced. In addition, an abstraction of databases of $\mathcal{M}^{\approx}$ is presented in terms of both type constraints and equality constraints. In the presence of the type and equality constraints, word constraint implication is investigated in Section 10.3. In this context, the decidability of the implication and finite implication problems for word constraints is established.

10.1 Complex value equality

In the previous chapters we have studied path constraint implication for a variety of data models, ranging from semistructured data models ($\mathcal{SM}$, $\mathcal{DM}$) to object-oriented models ($\mathcal{M}$, $\mathcal{M}^+$ and $\mathcal{M}^+_f$). In these models, the equality relation on data entities has a simple semantics.

As in OEM [8, 68, 70], it is assumed in $\mathcal{SM}$ and $\mathcal{DM}$ that each data entity has a unique

identity, and two data entities are equal if and only if they have the same identify. This equality relation is referred to as *identity equality*.

As in the object-oriented model studied in [3], the equality relation in $\mathcal{M}$, $\mathcal{M}^+$ and $\mathcal{M}_f^+$ is also defined to be identity equality. To illustrate this, consider a schema Δ in $\mathcal{M}$, $\mathcal{M}^+$ or $\mathcal{M}_f^+$. Let $\Delta = (\mathcal{C}, \nu, DBtype)$. Then as mentioned in Section 7.1, the set of types determined by Δ consists of $DBtype$, base types and class types. That is,

$$T(\Delta) \subseteq \mathcal{C} \cup \mathcal{B} \cup \{DBtype\}.$$

Each value of a type in $T(\Delta)$ has a unique identity. More specifically, an object of a class has an oid (object identity), a basic value (i.e., a value of a base type) is uniquely identified by the value itself, and in any instance of Δ, $DBtype$ has a distinguished value called the root. The equality relation on these values is defined by comparing their identities. As a result, a database of Δ can be viewed as a collection of identities. The value represented by an identity is either a basic value, a record consisting of basic values and oids, or a set consisting of basic values or oids. In other words, these models do not support complex values with nested structures such as sets of records or records with set components.

In some object-oriented database systems such as those studied in [5, 6, 31, 58], however, complex values with nested structures are common. For examples, recall the student/course schema given in Chapter 1. In this schema, the classes *student* and *course* are mapped to the following types:

$$\begin{aligned} student &\mapsto [Name : string, Taking : \{course\}] \\ course &\mapsto [CName : string, Enrolled : \{student\}] \end{aligned}$$

Values of such types are records with set components. These values are not allowed in $\mathcal{M}$, $\mathcal{M}^+$ or $\mathcal{M}_f^+$. Such a complex value may not have a unique identity. Consequently, the equality relation on complex values cannot be simply treated as identity equality.

The equality relation on complex values is called *complex value equality*, denoted by $\approx$ and defined as follows:

1. Let v_1 and v_2 be values of a base type. Then $v_1 \approx v_2$ if $v_1 = v_2$ (i.e., v_1 and v_2 are identical).

2. Let o_1 and o_2 be objects of a class. Then $o_1 \approx o_2$ if o_1 and o_2 have the same oid. That is, equality on objects is defined by comparing object identities.

3. Let s_1 and s_2 be of a set type. Then $s_1 \approx s_2$ if for every $x \in s_1$, there is $y \in s_2$ such that $x \approx y$, and vice versa.

4. Let r_1 and r_2 be values of record type $[l_1 : \tau_1, \ldots, l_n : \tau_n]$. Then $r_1 \approx r_2$ if for every $i \in [1, n]$, $r_1.l_i \approx r_2.l_i$. Here $v.l$ stands for the projection of v at attribute l.

A natural question to answer here is whether complex value equality interacts with path constraints. In other words, when complex value equality is supported, whether the complexity results on path constraint implication established in the previous chapters still hold? One may question the need to investigate this since at first glance, complex value equality appears to have no severe impact on path constraint implication. This appearance can be dispelled by considering the following example.

Example 10.1: We consider an object-oriented model, $\mathcal{M}^{\approx}$, which supports complex values with nested structures. That is, in $\mathcal{M}^{\approx}$, complex values such as records with set components are allowed. We compare $\mathcal{M}^{\approx}$ and $\mathcal{M}_f^+$ with respect to word constraint implication.

Let Δ be a schema in $\mathcal{M}_f^+$ and Δ' a schema in $\mathcal{M}^{\approx}$:

```
Δ:  class player {
          record (Name: string,
                  Rank: int)
    }
    class team {
          set(player)
    }
    Team1, Team2:  team;

Δ′: class player    /* as defined above
    Team1, Team2:  set(player);
```

An instance of Δ is a value of database type ($DBtype$):

$$[Team1 : team,\ Team2 : team].$$

Note that $team$ is a class type. Similarly, a database instance of Δ' is a value of $DBtype$:

$$[Team1 : \{player\},\ Team2 : \{player\}],$$

which is a complex value with nested structure. It should be noted that Δ' is not a schema in $\mathcal{M}_f^+$.

Let `Jones` and `Smith` be objects of `player`, created by:

```
Jones = new player("Jones", 2);
Smith = new player("Smith", 3);
```

Using these objects, an instance I of Δ and an instance I' of Δ' are given as follows:

I:
```
Team1 = new team(Jones, Smith);
Team2 = new team(Jones, Smith);
```

I':
```
Team1 = set(Jones, Smith);
Team2 = set(Jones, Smith);
```

The databases I and I' are represented by the graphs shown in Figure 10.1, in which r denotes the root and $*$ denotes the set membership relation.

Now let us consider the following word constraints:

$$\begin{aligned}
\varphi_1 &= \forall x\,(\exists y\,(Team1(r,y) \wedge *(y,x)) \rightarrow \exists y\,(Team2(r,y) \wedge *(y,x))) \\
\varphi_2 &= \forall x\,(\exists y\,(Team2(r,y) \wedge *(y,x)) \rightarrow \exists y\,(Team1(r,y) \wedge *(y,x))) \\
\varphi &= \forall x\,(Team1(r,x) \rightarrow Team2(r,x))
\end{aligned}$$

Here φ_1 and φ_2 ensure that `Team1` and `Team2` consist of the same players. Constraint φ in fact states that `Team1` and `Team2` are equal, since $DBtype$ is a record type in both Δ and Δ'. It is easy to see that $I \models \varphi_1 \wedge \varphi_2$ but $I \not\models \varphi$. This is because in I, `Team1` and `Team2` are objects with different oids even if they consist of the same players, i.e., they

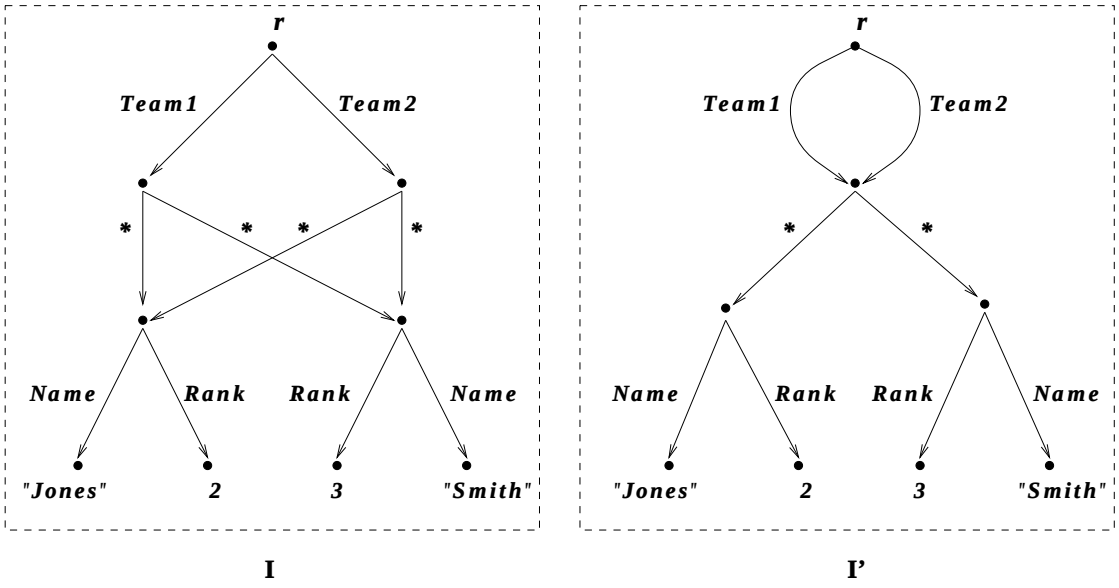

Figure 10.1: Databases in Example 10.1

have the same value. However, $I' \models \varphi_1 \wedge \varphi_2$ and $I' \models \varphi$. The reason is that in I', `Team1` and `Team2` are defined to be sets. As a result, they are equal as long as they consist of the same players. In fact, it can be shown that in the context of database instances of Δ', $\{\varphi_1, \varphi_2\}$ indeed implies φ. In contrast, over any schema Δ in $\mathcal{M}_f^+$, if φ_1, φ_2 and φ are in $P_w(\Delta)$, then $\{\varphi_1, \varphi_2\}$ does not imply φ.

This example shows that the difference between the definitions of equality in $\mathcal{M}_f^+$ and $\mathcal{M}^{\approx}$ gives rise to different outcomes of word constraint implication. ■

As illustrated by the example above, complex value equality interacts with path constraints. In other words, it does make life harder. In the context of $\mathcal{M}_f^+$, $\mathcal{M}^+$ and $\mathcal{M}$, it has been shown in Chapter 7 that a schema can be characterized in terms of a type constraint. The interaction between type constraints and path constraints has been studied in Chapter 9. However, in the presence of complex value equality, an equality constraint is also needed, in addition to type constraint, to capture the semantics of a schema. The interaction of the three different forms of constraints – equality, type and path constraints – has not been addressed in the previous chapters or reported in any literature.

The rest of this chapter investigates the interaction of equality, type and word constraints.

To focus on the central issue and to simplify the discussion, we define an object-oriented model $\mathcal{M}^{\approx}$ in the flavor of the nested relational model [5]. That is, the set and record constructs are required to alternate (i.e., set of sets and record with a record component are prohibited). We characterize a schema Δ in $\mathcal{M}^{\approx}$ in terms of an equality constraint, in addition to the type constraint defined in Chapter 7. We represent database instances of Δ as (finite) logic structures satisfying both the type constraint and the equality constraint. Using this abstraction, we show that the implication and finite implication problems for word constraints are decidable in the context of $\mathcal{M}^{\approx}$. More specifically, we present an elementary proof of the decidability by giving a small model argument. That is, given a finite set $\Sigma \cup \{\varphi\}$ of word constraints, we show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model, then it has a model of size at most exponential in the length of $\bigwedge \Sigma \wedge \neg\varphi$.

It should be noted that complex value equality is a notion different from the so called shallow and deep equalities [3]. To clarify the difference, below we review the definitions of these predicates.

- *Shallow equality*, denoted by $\sim_s$, is defined in the same way as value equality except for the case of objects, which is given as follows. Let o_1 and o_2 be objects of a class, and v_1, v_2 be the values of o_1 and o_2, respectively. Then $o_1 \sim_s o_2$ if $v_1 \approx v_2$.
- *Deep equality*, denoted by $\sim_d$, can be described as follows.
  1. Let v_1 and v_2 be values of an atomic type. Then $v_1 \sim_d v_2$ if $v_1 = v_2$ (i.e., v_1 and v_2 are identical).
  2. Let o_1 and o_2 be objects of a class, and v_1, v_2 be the values of o_1 and o_2, respectively. Then $o_1 \sim_d o_2$ if $v_1 \sim_d v_2$.
  3. Let s_1 and s_2 be of a set type. Then $s_1 \sim_d s_2$ if for every $x \in s_1$, there is $y \in s_2$ such that $x \sim_d y$, and vice versa.
  4. Let r_1 and r_2 be values of record type $[l_1 : \tau_1, \ldots, l_n : \tau_n]$. Then $r_1 \sim_d r_2$ if for every $i \in [1, n]$, $r_1.l_i \sim_d r_2.l_i$.

Shallow and deep equalities on objects are not oid equality. That is, they are not defined by comparing object identities. In particular, the definition of deep equality is recursive

and thus is not definable in first-order logic [3]. In contrast, we shall show that complex value equality is definable in first-order logic.

10.2 The object-oriented model $\mathcal{M}^{\approx}$

In this section, we present the object-oriented model $\mathcal{M}^{\approx}$, define equality constraints, and give an abstraction of the databases of $\mathcal{M}^{\approx}$ in terms of equality and type constraints. Finally, we justify the abstraction with respect to word constraint implication.

10.2.1 Schemas and instances in $\mathcal{M}^{\approx}$

We begin by describing the object-oriented model $\mathcal{M}^{\approx}$. Assume a fixed countable set of labels, $\mathcal{L}$, and a fixed finite set of *base types*, $\mathcal{B}$.

Definition 10.1: Let $\mathcal{C}$ be some finite set of *classes*. The set of *types over* $\mathcal{C}$, denoted by $Types^{\mathcal{C}}$, is defined by the following abstract syntax:

$$\begin{aligned} \tau &::= b \mid C \mid Set \mid Record \\ Set &::= \{b\} \mid \{C\} \\ Record &::= [l_1 : \delta, \ldots, l_n : \delta] \\ \delta &::= b \mid Set \end{aligned}$$

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$. We reserve τ to range over $Types^{\mathcal{C}}$. ■

Definition 10.2: A *schema* in $\mathcal{M}^{\approx}$ is a triple $\Delta = (\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ is a finite set of classes,
- ν is a mapping: $\mathcal{C} \rightarrow Types^{\mathcal{C}}$ such that for each $C \in \mathcal{C}$, $\nu(C) \in Record$, and
- $DBtype$ is a record type in $Types^{\mathcal{C}}$ having the form:

$$[l_1 : \{\tau_1\}, \ldots, l_n : \{\tau_n\}]$$

■

Here we assume that every database has a unique (persistent) entry point, and $DBtype$ in a schema specifies the type of the entry point. The entry point is a collection of sets. This definition of database schema is in the same spirit as those found in [5, 6, 58].

The following should be noted about the definitions above.

- Given a schema $\Delta = (\mathcal{C}, \nu, DBtype)$ in $\mathcal{M}^{\approx}$, $DBtype$ can be converted into a "pure type", i.e., a type containing no classes, by continuously substituting $\nu(C)$ for C for any $C \in \mathcal{C}$. In this pure type, the set and record constructs alternate. That is, there is no set of sets or record with a record component. This is in the flavor of the nested relational model [5].

- To simplify the discussion, we require that a set consists of either basic values or oids. That is, in $\mathcal{M}^{\approx}$, complex values with nested structures are limited to be values of *Record*. In addition, a record in $\mathcal{M}^{\approx}$ consists of only basic values and sets. This restriction can be removed without affecting the decidability results of this chapter.

Example 10.2: The schema Δ' given in Example 10.1 can be described as $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C} = \{player\}$,
- ν maps $player$ to $[Name : string, Rank : int]$, and
- $DBtype = [Team1 : \{player\}, Team2 : \{player\}]$.

■

Example 10.3: The student/course schema given in Chapter 1 can be specified in $\mathcal{M}^{\approx}$ as $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C} = \{student, course\}$,
- ν is defined by:

$$\begin{aligned} student &\mapsto [Name : string, Taking : \{course\}] \\ course &\mapsto [CName : string, Enrolled : \{student\}] \end{aligned}$$

- $DBtype = [Students : \{student\},\ Courses : \{course\}]$.

It should be noted that this is not a schema in $\mathcal{M}_f^+$, $\mathcal{M}^+$ or $\mathcal{M}$. ■

The instances of a schema in $\mathcal{M}^{\approx}$ is defined in the same way as in Definition 7.3.

Example 10.4: The instance I' of Δ' given in Example 10.1 can be described as $(\pi,\ \mu,\ d)$, where

- $\pi(player) = \{Jones, Smith\}$,
- $\mu : player \rightarrow [\![[Name : string,\ Rank : int]]\!]_\pi$ is defined by

$$\begin{aligned} Jones &\mapsto [Name : \text{“Jones”},\ Rank : 2] \\ Smith &\mapsto [Name : \text{“Smith”},\ Rank : 3] \end{aligned}$$

- d is defined to be $[Team1 : \{Jones, Smith\},\ Team2 : \{Jones, Smith\}]$.

■

10.2.2 Equality and type constraints

As in Chapter 7, we present an abstraction of databases of $\mathcal{M}^{\approx}$ in terms of first-order logic. We first characterize every schema Δ in $\mathcal{M}^{\approx}$ by means of two first-order logic sentences, called *the type constraint* and *the equality constraint determined by* Δ, respectively. We then represent databases of Δ as (finite) logic structures satisfying the type and equality constraints. Such a structure can be depicted as an edge-labeled rooted directed graph.

To define type and equality constraints, we first specify the first-order vocabulary determined by a schema. As in Definition 7.4, given a schema Δ in $\mathcal{M}^{\approx}$, the set of binary relation symbols $E(\Delta)$ and the set of types $T(\Delta)$ determined by Δ can be defined. Using $T(\Delta)$ and $E(\Delta)$, we define the vocabulary as follows.

Definition 10.3: The *signature determined by a schema* Δ *in* $\mathcal{M}^{\approx}$ is a quadruple

$$\sigma(\Delta) = (r,\ E(\Delta),\ R(\Delta),\ Q(\Delta)),$$

where

- r is a constant symbol, denoting the root;
- $E(\Delta)$ is the finite set of binary relation symbols denoting the edge labels;
- $R(\Delta)$ is the finite set of unary relation symbols defined by $\{R_\tau \mid \tau \in T(\Delta)\}$, which denote the sorts; and
- $Q(\Delta)$ is the finite set of binary relation symbols defined by $\{\approx_\tau \mid \tau \in T(\Delta)\}$, which denote the equality symbols of different sorts.

■

Example 10.5: The signature determined by the schema of Example 10.3 is $(r,\ E,\ R,\ Q)$, where

- r is a constant, which in each instance $(\pi,\ \mu,\ d)$ of the schema intends to name d;
- $E = \{*,\ \mathit{Students},\ \mathit{Courses},\ \mathit{Name},\ \mathit{Taking},\ \mathit{CName},\ \mathit{Enrolled}\}$;
- $R = \{R_\tau \mid \tau \in T\}$ and $Q = \{\approx_\tau \mid \tau \in T\}$, where

$$T = \{\mathit{DBtype},\ \mathit{student},\ \mathit{course},\ \mathit{string},\ \{\mathit{student}\},\ \{\mathit{course}\}\}.$$

■

Given the vocabulary $\sigma(\Delta)$ determined by schema Δ, we define the type constraint $\Phi(\Delta)$ determined by Δ in the same way as in Definition 7.6. In addition, we define the equality constraint determined by Δ as follows.

Definition 10.4: Let Δ be a schema in $\mathcal{M}^{\approx}$. For every τ in $T(\Delta)$, the *equality constraint determined by* τ is the sentence $\forall\, x\, y\, \phi^{\approx}_\tau(x, y)$ defined as follows.

- If $\tau = b$ or $\tau = C$ for some $C \in \mathcal{C}$, then $\phi^{\approx}_\tau(x, y)$ is

$$x \approx_\tau y \ \leftrightarrow\ x = y.$$

- If $\tau = \{\tau'\}$, then $\phi^{\approx}_\tau(x, y)$ is

$$x \approx_\tau y \ \leftrightarrow\ R_\tau(x) \wedge R_\tau(y) \wedge\ \forall\, z_1\, \exists\, z_2\, (*(x, z_1) \rightarrow *(y, z_2) \wedge z_1 \approx_{\tau'} z_2) \wedge$$
$$\forall\, z_1\, \exists\, z_2\, (*(y, z_1) \rightarrow *(x, z_2) \wedge z_1 \approx_{\tau'} z_2).$$

- If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $\phi^{\approx}_{\tau}(x, y)$ is

$$x \approx_\tau y \;\leftrightarrow\; R_\tau(x) \wedge R_\tau(y) \wedge \bigwedge_{i \in [1,n]} \forall z_1 \forall z_2 \, (l_i(x, z_1) \wedge l_i(y, z_2) \rightarrow z_1 \approx_{\tau_i} z_2).$$

The *equality constraint determined by* Δ is the sentence $\Phi^{\approx}(\Delta)$ defined by

$$\bigwedge_{\tau \in T(\Delta)} \forall x \, y \, \phi^{\approx}_{\tau}(x, y) \;\wedge\; \forall x \, y \, (\bigvee_{\tau \in T(\Delta)} x \approx_\tau y \;\rightarrow\; x = y).$$

■

The following should be noted.

- For each $\tau \in T(\Delta)$, the definition of $\approx_\tau$ is not recursive, i.e., $\approx_\tau$ is not defined in terms of itself. This is because in $\mathcal{M}^{\approx}$, only classes can be defined recursively, and the equality on objects is defined by comparing oids.
- The type constraint is in two-variable logic with counting, C^2. In contrast, the equality constraint cannot be expressed in C^2.
- Both type and equality constraints are definable in first-order logic.

Using the type and equality constraints defined above, we give an abstraction of the databases of $\mathcal{M}^{\approx}$ as follows.

Definition 10.5: An *abstract database of a schema* Δ is a finite $\sigma(\Delta)$-structure G such that $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$. We denote the set of all abstract databases of schema Δ by $\mathcal{U}_f(\Delta)$.

We use $\mathcal{U}(\Delta)$ to denote the set of all the $\sigma(\Delta)$-structures satisfying the following conditions: for each $G \in \mathcal{U}(\Delta)$,

- $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$; and
- G respects *the finite set rule*. That is, for each set type $\tau \in T(\Delta)$ and for each $o \in R^G_\tau$, there are only finitely many o' in G such that $G \models *(o, o')$.

  As a result, each node in G has finitely many outgoing edges.

■

It should be noted that $\mathcal{U}(\Delta)$ is not definable in first-order logic.

The justification of the abstraction will be given later in this section.

Next, we compare $\mathcal{M}^{\approx}$ with $\mathcal{M}_f^+$.

In contrast to $\mathcal{M}^{\approx}$, $\mathcal{M}_f^+$ does not support complex values with nested structures. In addition, the equality relation in $\mathcal{M}_f^+$ is defined to be identity equality. That is, two data entities are equal if and only if they have the same identity.

Because the equality in $\mathcal{M}_f^+$ has a simple semantics, a schema Δ in $\mathcal{M}_f^+$ can be characterized by the type constraint $\Phi(\Delta)$ alone. As a result, an abstract database of Δ is defined to be a finite $\sigma(\Delta)$-structure satisfying $\Phi(\Delta)$, and $\mathcal{U}(\Delta)$ is defined to be the set of $\sigma(\Delta)$-structures which satisfy $\Phi(\Delta)$ and respect the finite set rule.

10.2.3 Word constraints revisited

As in the context of $\mathcal{M}_f^+$, given a schema Δ in $\mathcal{M}^{\approx}$, the notions of $Paths(\Delta)$, $P_w(\Delta)$, $\models_\Delta$, $\models_{(f,\Delta)}$, and the implication and finite implication problems for $P_w(\Delta)$ over Δ can be defined. Similarly, the implication and finite implication problems for word constraints in the context of $\mathcal{M}^{\approx}$ can also be defined (see Section 7.1 for these definitions).

It should be mentioned that over a schema Δ in $\mathcal{M}^{\approx}$, the implication and (finite) implication problem for $P_w(\Delta)$ is considered in connection with $\mathcal{U}(\Delta)$ ($\mathcal{U}_f(\Delta)$), which consists of $\sigma(\Delta)$-structures satisfying both $\Phi(\Delta)$ and $\Phi^{\approx}(\Delta)$. As mentioned above, $\Phi^{\approx}(\Delta)$ is not definable in C^2. Therefore, word constraint implication cannot be stated in C^2 in the context of $\mathcal{M}^{\approx}$. Recall that in Section 8.1, we establish the decidability of the finite implication problem for word constraints in the context of $\mathcal{M}_f^+$ by reducing it to the finite satisfiability problem for C^2. However, in the context of $\mathcal{M}^{\approx}$, the proof is no longer applicable. In the next section, we shall investigate word constraint implication in the context of $\mathcal{M}^{\approx}$.

Next, we justify the abstraction of databases of $\mathcal{M}^{\approx}$ defined above.

Example 10.6: The following are word constraints over the schema given in Example 10.3.

$$\begin{aligned} \phi &= \forall x\,(Students \cdot * \cdot Taking \cdot *(r,x) \rightarrow Courses \cdot *(r,x)), \\ \varphi &= \forall x\,(Courses \cdot * \cdot Enrolled \cdot *(r,x) \rightarrow Students \cdot *(r,x)). \end{aligned}$$

In an instance $(\pi,\ \mu,\ d)$ of the schema, ϕ and φ are interpreted as:

$$\begin{aligned} \forall x\,(\exists y\,(y \in d.Students \wedge x \in y.Taking) &\rightarrow x \in d.Courses) \\ \forall x\,(\exists y\,(y \in d.Courses \wedge x \in y.Enrolled) &\rightarrow x \in d.Students) \end{aligned}$$

respectively. As mentioned earlier, $v.l$ stands for the projection of record v at attribute l. The constraint ϕ states: "any course taken by a student in database d is a course in d", and φ states: "any student who enrolls in a course in database d is a student in d". ∎

As illustrated by Example 10.6, word constraints over a schema Δ in $\mathcal{M}^{\approx}$ can be naturally interpreted in database instances of Δ. Likewise, the notion "$I \models \varphi$" can also be defined for an instance I of Δ and a constraint φ of $P_w(\Delta)$.

The agreement between databases of $\mathcal{M}^{\approx}$ and their abstraction with respect to word constraint implication is elaborated by the following lemma. This lemma is similar to Lemma 7.1 established in the context of M_f^+.

Lemma 10.1: Let Δ be any schema in $\mathcal{M}^{\approx}$. Then for each $I \in \mathcal{I}(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$, such that

$$(\dagger) \qquad \text{for every } \varphi \in P_w(\Delta),\ \ I \models \varphi \ \text{ iff } \ G \models \varphi$$

Similarly, for each $G \in \mathcal{U}_f(\Delta)$, there is $I \in \mathcal{I}(\Delta)$, such that $(\dagger)$ holds. ∎

Proof: Let $\Delta = (\mathcal{C},\ \nu,\ DBtype)$.

(1) Given $I \in \mathcal{I}(\Delta)$, we construct $G \in \mathcal{U}_f(\Delta)$, such that for each $\varphi \in P_w(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

Let $I = (\pi,\ \mu,\ d)$. Then we define V to be the smallest set satisfying the following:

1. $d \in V$;

2. for every $v \in V$,

   - if v is a set, then every element of v is in V;
   - if v is a record (or v is an object and $\mu(v)$ is a record), then every attribute of v (or $\mu(v)$) is in V.

For every $v \in V$, let $o(v)$ be a distinct node. Let $G = (|G|, r^G, E^G, R^G, Q^G)$, where

- $|G| = \{o(v) \mid v \in V\}$;
- $r^G = o(d)$;
- for each $o(v) \in |G|$ and $\tau \in T(\Delta)$, $G \models R^G_\tau(o(v))$ iff v is of type τ;
- for each $\tau \in T(\Delta)$, the relation $\approx_\tau$ is defined to be $\{(o(v), o(v)) \mid o(v) \in R^G_\tau\}$;
- for all $o(v), o(v') \in |G|$,
  - for each $l \in \mathcal{L} \cap E(\Delta)$, $G \models l(o(v), o(v'))$ iff $v' = v.l$ (or $v' = \mu(v).l$ if v is an object);
  - $G \models *(o(v), o(v'))$ iff $v' \in v$.

Then it is straightforward to verify the following:

- $G \in \mathcal{U}_f(\Delta)$; that is, G is a finite $\sigma(\Delta)$-structure and $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$;
- for each $\varphi \in P_w(\Delta)$, $G \models \varphi$ iff $I \models \varphi$. This can be easily verified by *reductio*.

(2) Given $G = (|G|, r^G, E^G, R^G, Q^G)$ in $\mathcal{U}_f(\Delta)$, we define $I = (\pi, \mu, d)$ in $\mathcal{I}(\Delta)$, such that for every $\varphi \in P_w(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

To simplify the discussion, we assume that for every base type b, its domain D_b is infinite (a similar proof for the finite case can be given as in the proof of Lemma 7.1). By this assumption, there exists an injective mapping $g_b : R^G_b \to D_b$, where R^G_b is the unary relation in G denoting the sort b.

For every $C \in \mathcal{C}$, let $\pi(C) = R_C^G$. We then define a mapping

$$f : |G| \to \bigcup_{\tau \in T(\Delta)} [\![\tau]\!]_\pi$$

as follows: For each $o \in |G|$,

- if $o \in R_C^G$ for some $C \in \mathcal{C}$, then let $f(o) = o$;
- if $o \in R_b^G$ for some base type b, then let $f(o) = g(o)$;
- if $o \in R_\tau^G$ and $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then let $f(o) = [l_1 : f(o_1), \ldots, l_n : f(o_n)]$, where for $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$;
- if $o \in R_\tau^G$ and $\tau = \{\tau'\}$, then let $f(o) = \{f(o') \mid o' \in |G|,\ G \models *(o, o')\}$.

Note that f is well-defined and is an injection, since G is finite and $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$. Now let

- $d = f(r^G)$;
- for each $C \in \mathcal{C}$ and each $o \in \pi(C)$, if $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then

  $$\mu(o) = [l_1 : f(o_1), \ldots, l_n : f(o_n)],$$

  where for $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$.

Again, this is well-defined. In addition, it is easy to verify that $I \in \mathcal{I}(\Delta)$, and moreover, $G \models \varphi$ iff $I \models \varphi$. ∎

From Lemma 10.1 follows immediately the corollary below.

Corollary 10.2: Let Δ be any schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be any finite subset of $P_w(\Delta)$. Then there is $I \in \mathcal{I}(\Delta)$ such that $I \models \bigwedge \Sigma \wedge \neg\varphi$ if and only if there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

10.3 Word constraint implication in the context of $\mathcal{M}^{\approx}$

In this section, we establish the decidability of the implication and finite implication problems for word constraints in the context of $\mathcal{M}^{\approx}$.

Theorem 10.3: Over arbitrary schema Δ in $\mathcal{M}^{\approx}$, both the implication and finite implication problems for $P_w(\Delta)$ are decidable. ∎

We prove Theorem 10.3 by giving a small model argument. Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$. Given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, we show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model G of size at most 2^{mN}, and $G \in \mathcal{U}_f(\Delta)$, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the *maximum record width of* Δ defined by

$$m = 1 + max\{\ n \mid \tau \in T(\Delta),\ \tau = [l_1 : \tau_1,\ \ldots,\ l_n : \tau_n] \text{ or } \nu(\tau) = [l_1 : \tau_1,\ \ldots,\ l_n : \tau_n] \text{ if } \tau \in \mathcal{C}\}.$$

Before we give the proof, we first describe two techniques used to establish the small model property. The first is a simple filtration argument. Using it we show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a $\sigma(\Delta)$-structure G such that the size of G is at most 2^{mN} and

$$G \models \Sigma \wedge \neg\varphi \wedge \Phi(\Delta).$$

It should be noted that if the equality constraint $\Phi^{\approx}(\Delta)$ is not taken into account, this filtration argument alone suffices. The second technique is referred to as *identifying* operation. Using it we construct H from G such that the size of H is no larger than the size of G, and in addition, under certain conditions,

$$H \models \Sigma \wedge \neg\varphi \wedge \Phi(\Delta) \wedge \Phi^{\approx}(\Delta).$$

Finally, we use a slightly stronger filtration argument and the identifying operation to prove Theorem 10.3.

10.3.1 A filtration argument

We first present a simple filtration argument. It should be noted that for an object-oriented model which does not support complex values with nested structures and is in the flavor of the nested relational model, this argument alone is sufficient to establish the small model property for word constraint implication.

We begin with a few definitions.

Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. Then we define the following:

$$\begin{aligned} Pts(\Sigma \cup \{\varphi\}) &= \{lt(\phi), rt(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\} \\ CloPts(\Sigma \cup \{\varphi\}) &= \{\rho \mid \varrho \in Pts(\Sigma \cup \{\varphi\}),\ \rho \preceq_p \varrho\} \end{aligned}$$

Here the notations lt and rt are described in Definition 7.10, and $\rho \preceq_p \varrho$ stands for that ρ is a prefix of ϱ, as described in Chapter 2.

It is straightforward to verify the following.

Lemma 10.4: Let N be the length of $\bigwedge \Sigma \wedge \neg\varphi$. Then the cardinality of $CloPts(\Sigma \cup \{\varphi\})$ is at most N. ∎

Let G be a $\sigma(\Delta)$-structure. Then for every $a \in |G|$, we define

$$lb(a,\, G,\, \Sigma \cup \{\varphi\}) = \{\rho \mid \rho \in CloPts(\Sigma \cup \{\varphi\}),\ G \models \rho(r^G,\, a)\}.$$

In addition, we define *the label of G with respect to* $\Sigma \cup \{\varphi\}$ to be

$$LB(G,\, \Sigma \cup \{\varphi\}) = \{lb(a,\, G,\, \Sigma \cup \{\varphi\}) \mid a \in |G|\}.$$

An important property of $LB(G,\, \Sigma \cup \{\varphi\})$ is that it characterizes whether $G \models \bigwedge \Sigma \wedge \neg\varphi$.

Lemma 10.5: Suppose that $\bigwedge \Sigma \wedge \neg\varphi$ has a model G. Then for every $\sigma(\Delta)$-structure H, if $LB(H,\, \Sigma \cup \{\varphi\}) = LB(G,\, \Sigma \cup \{\varphi\})$, then $H \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

Proof: We first show that $H \models \Sigma$. Suppose, for *reductio* , that there is $\phi \in \Sigma$ and $a \in |H|$, such that

$$H \models lt(\phi)(r^H,\, a) \wedge \neg rt(\phi)(r^H,\, a).$$

Then we have $lt(\phi) \in lb(a, H, \Sigma \cup \{\varphi\})$, but $rt(\phi) \notin lb(a, H, \Sigma \cup \{\varphi\})$. By the assumption that $LB(H, \Sigma \cup \{\varphi\}) = LB(G, \Sigma \cup \{\varphi\})$, we have $lb(a, H, \Sigma \cup \{\varphi\}) \in LB(G, \Sigma \cup \{\varphi\})$. Hence there is $b \in |G|$ such that $lb(b, G, \Sigma \cup \{\varphi\}) = lb(a, H, \Sigma \cup \{\varphi\})$. Therefore,

$$G \models lt(\phi)(r^G, b) \wedge \neg rt(\phi)(r^G, b).$$

Hence $G \not\models \phi$. This contradicts the assumption that $G \models \Sigma$.

Next, we show that $H \models \neg\varphi$. By $G \models \neg\varphi$, there exists $b \in |G|$ such that

$$G \models lt(\varphi)(r^G, b) \wedge \neg rt(\varphi)(r^G, b).$$

Therefore, we have $lt(\varphi) \in lb(b, G, \Sigma \cup \{\varphi\})$, but $rt(\varphi) \notin lb(b, G, \Sigma \cup \{\varphi\})$. By the assumption that $LB(H, \Sigma \cup \{\varphi\}) = LB(G, \Sigma \cup \{\varphi\})$, there must be some node $a \in |H|$ such that $lb(a, H, \Sigma \cup \{\varphi\}) = lb(b, G, \Sigma \cup \{\varphi\})$. Therefore,

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a).$$

Hence $H \models \neg\varphi$. ∎

An important property of the type constraints $\Phi(\Delta)$ is described as follows.

Lemma 10.6: Let Δ be a schema in $\mathcal{M}^{\approx}$ and α be a path in $Paths(\Delta)$. Then for every $\sigma(\Delta)$-structure G satisfying $\Phi(\Delta)$ and for every node o in G, if $G \models \alpha(r^G, o)$, then $o \in R^G_\tau$, where $\tau = type(\alpha)$. ∎

Proof: A straightforward induction on $|\alpha|$. ∎

Next, we present the filtration argument.

Proposition 10.7: Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model G and $G \models \Phi(\Delta)$, then there is a $\sigma(\Delta)$-structure H such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta),$$

and the size of H is at most 2^{mN}, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ. ∎

Proof: Let $\Delta = (\mathcal{C}, \nu, DBtype)$. Since $G \models \Phi(\Delta)$, for every $a \in |G|$, there is $\tau \in T(\Delta)$ such that $a \in R^G_\tau$. We define $mlb(a)$ to be either

- $lb(a,\ G,\ \Sigma \cup \{\varphi\})$, if $\tau \notin \mathcal{C}$ and τ is not a record type, or
- $(lb(a,\ G,\ \Sigma \cup \{\varphi\}),\ (l_1,\ lb(a_1,\ G,\ \Sigma \cup \{\varphi\})),\ \ldots,\ (l_n,\ lb(a_n,\ G,\ \Sigma \cup \{\varphi\})))$, if τ is record type $[l_1 : \tau_1,\ \ldots,\ l_n : \tau_n]$ (or $\tau \in \mathcal{C}$ and $\nu(\tau) = [l_1 : \tau_1,\ \ldots,\ l_n : \tau_n]$), and for $i \in [1, n]$, $G \models l_i(a,\ a_i)$.

Let $MLB(G) = \{mlb(a) \mid a \in |G|\}$. For each $s \in MLB(G)$, let o_s be a distinct node. We define a function $f : |G| \to MLB(G)$ such that for each $a \in |G|$, $f : a \mapsto o_s$ where $s = mlb(a)$.

Using f, we define $H = (|H|,\ r^H,\ E^H,\ R^H,\ Q^H)$ as follows.

- $|H| = \{f(a) \mid a \in |G|\}$.
- $r^H = f(r^G)$.
- E^H is populated as follows: for each $K \in E$ and $o_a,\ o_b \in |H|$, $H \models K(o_a,\ o_b)$ iff there exist $a, b \in |G|$ such that $G \models K(a,\ b)$, $f(a) = o_a$ and $f(b) = o_b$.
- For every $\tau \in T(\Delta)$ and $o \in |H|$, $o \in R^H_\tau$ iff there is $a \in |G|$ such that $f(a) = o$ and $a \in R^G_\tau$. This is well-defined by Lemma 10.6 and the assumption that $G \models \Phi(\Delta)$.
- For every $\tau \in T(\Delta)$, we define $\approx_\tau$ be $\{(o,\ o) \mid o \in R^H_\tau\}$.

Next, we show that H is indeed the structure described in the proposition.

(1) The size of H.

For every $a \in |G|$, either $mlb(a) \subseteq CloPts(\Sigma \cup \{\varphi\})$, or $mlb(a) \subseteq CloPts^m(\Sigma \cup \{\varphi\})$. Hence by Lemma 10.4, the size of H is at most 2^{mN}.

(2) $H \models \Phi(\Delta)$.

By Lemma 10.6 and the assumption that $G \models \Phi(\Delta)$, it is easy to verify that for every $o \in |H|$, there is a unique $\tau \in T(\Delta)$ such that $o \in R^H_\tau$. In addition, by the definition of H, it is easy to verify the following.

- $r^H \in R^H_{DBtype}$.

- For every $\tau \in T(D)$ and $o \in R^H_\tau$,
  - if τ is either a base type or a set type, then $H \models \phi_\tau(o)$;
  - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ for some $C \in \mathcal{C}$, then

$$H \models \forall\, y\, (\bigwedge_{l \in E(\Delta) \setminus \{l_1, \ldots, l_n\}} \neg l(o, y)) \ \wedge \bigwedge_{i \in [1,n]} (\exists\, y\, l_i(o, y) \ \wedge \forall\, y\, (l_i(o, y) \rightarrow R_{\tau_i}(y))).$$

By the definition of $mlb(o)$, it can also be shown that if τ or $\nu(\tau)$ is a record type, then

$$H \models \bigwedge_{i \in [1,n]} \exists\, !\, y\, l_i(o, y).$$

To see this, without loss of generality, suppose for *reductio* that there exist $o_1, o_2 \in |H|$, such that

$$H \models l_1(o,\, o_1) \wedge l_1(o,\, o_2) \wedge o_1 \neq o_2.$$

Then by the definition of E^H, there exist $a, b, c, d \in |G|$ such that $f(a) = f(c) = o$, $f(b) = o_1$, $f(d) = o_2$, and $G \models l_1(a,\, b) \wedge l_1(c,\, d)$. By the definition of f and mlb, we have:

$$\begin{aligned}
mlb(a) &= (lb(a,\, G,\, \Sigma \cup \{\varphi\}),\ (l_1,\, lb(b,\, G,\, \Sigma \cup \{\varphi\})),\ \ldots) \\
mlb(c) &= (lb(c,\, G,\, \Sigma \cup \{\varphi\}),\ (l_1,\, lb(d,\, G,\, \Sigma \cup \{\varphi\})),\ \ldots) \\
mlb(a) &= mlb(c) \\
mlb(b) &\neq mlb(d)
\end{aligned}$$

By Definition 10.1, τ_1 is neither a class type nor a record type. Hence

$$\begin{aligned}
mlb(b) &= lb(b,\, G,\, \Sigma \cup \{\varphi\}), \\
mlb(d) &= lb(d,\, G,\, \Sigma \cup \{\varphi\}).
\end{aligned}$$

Hence by $mlb(b) \neq mlb(d)$, we have $mlb(a) \neq mlb(c)$. This contradicts the assumption that $f(a) = f(c)$.

Therefore, we have $H \models \Phi(\Delta)$.

(2) $H \models \bigwedge \Sigma \wedge \neg\varphi$.

It suffices to show the following claim.

Claim: For every $a \in |G|$, $lb(a, G, \Sigma \cup \{\varphi\}) = lb(f(a), H, \Sigma \cup \{\varphi\})$.

For if the claim holds, then $LB(H, \Sigma \cup \{\varphi\}) = LB(G, \Sigma \cup \{\varphi\})$. Thus by Lemma 10.5, we have $H \models \bigwedge \Sigma \wedge \neg\varphi$.

Next, we show that for every $\alpha \in CloPts(\Sigma \cup \{\varphi\})$,

$$\alpha \in lb(a, G, \Sigma \cup \{\varphi\}) \quad \text{iff} \quad \alpha \in lb(f(a), H, \Sigma \cup \{\varphi\}).$$

This can be verified by induction on $|\alpha|$ as follows.

Base case: $\alpha = \epsilon$. Note that r^G is the unique node in G such that $\epsilon \in lb(r^G, G, \Sigma \cup \{\varphi\})$. Hence $\epsilon \in lb(a, G, \Sigma \cup \{\varphi\})$ iff $a = r^G$ iff $f(a) = r^H$ iff $\epsilon \in lb(r^H, H, \Sigma \cup \{\varphi\})$.

Inductive step: Assume the claim for $|\alpha|$. We next show that the claim also holds for $\alpha \cdot K$, where $\alpha \cdot K \in CloPts(\Sigma \cup \{\varphi\})$.

If $\alpha \cdot K \in lb(a, G, \Sigma \cup \{\varphi\})$, then there is $b \in |G|$ such that

$$G \models \alpha(r^G, b) \wedge K(b, a).$$

By the induction hypothesis, we have $\alpha \in lb(f(b), H, \Sigma \cup \{\varphi\})$. By the definition of E^H,

$$H \models K(f(b), f(a)).$$

Therefore, $\alpha \cdot K \in lb(f(a), H, \Sigma \cup \{\varphi\})$.

If $\alpha \cdot K \in lb(f(a), H, \Sigma \cup \{\varphi\})$, then there exists $b \in |H|$ such that

$$H \models \alpha(r^H, b) \wedge K(b, f(a)).$$

By the definition of E^H, there exist $o_1, o_2 \in |G|$ such that $f(o_1) = b$, $f(o_2) = f(a)$ and $G \models K(o_1, o_2)$. By the induction hypothesis, we have

$$\alpha \in lb(o_1, G, \Sigma \cup \{\varphi\}).$$

Hence $\alpha \cdot K \in lb(o_2, G, \Sigma \cup \{\varphi\})$. By $f(o_2) = f(a)$ and the definition of f, we have

$$lb(a, G, \Sigma \cup \{\varphi\}) = lb(o_2, G, \Sigma \cup \{\varphi\}).$$

Hence $\alpha \cdot K \in lb(a, G, \Sigma \cup \{\varphi\})$.

Therefore, the claim holds. This completes the proof of Proposition 10.7 ∎

10.3.2 Identifying operation

Next, we introduce an operation called *identifying*. Using this operation we are able to construct structures that satisfy necessary equality constraints. More specifically, let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. Then if there is a $\sigma(\Delta)$-structure G such that

$$G \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$$

and G satisfies certain conditions, then we can construct H from G such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta) \wedge \Phi^{\approx}(\Delta),$$

and the size of H is no larger than the size of G.

Before we describe the identifying operation, we first present some basic properties of the data model $\mathcal{M}^{\approx}$.

Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Delta = (\mathcal{C}, \nu, DBtype)$. Then we use $BC(\Delta)$ to denote the set $T(\Delta) \cap (\mathcal{B} \cup \mathcal{C})$.

Lemma 10.8: Let Δ be a schema in $\mathcal{M}^{\approx}$, $\Delta = (\mathcal{C}, \nu, DBtype)$ and G be a $\sigma(\Delta)$-structure such that $G \models \Phi(\Delta)$. Then G has the following properties.

1. For every $a \in |G|$, if $a \in R^G_\tau$ and $\tau \notin BC(\Delta)$, then either $\tau = DBtype$ or $\tau = \{\tau'\}$.
2. For every $a \in |G|$, if $a \in R^G_{DBtype}$ and $a \neq r^G$, then for every path $\alpha \in Paths(\Delta)$, $G \not\models \alpha(r^G, a)$.

■

Proof: By the definition of $T(\Delta)$ and $Paths(\Delta)$, it is easy to see that if $\tau \in T(\Delta)$, then there is $\alpha \in Paths(\Delta)$ such that $type(\alpha) = \tau$. Hence it suffices to show the following claim.

Claim: For every $\alpha \in Paths(\Delta)$, if $\alpha \neq \epsilon$, then either $type(\alpha) \in BC(\Delta)$ or $type(\alpha)$ is a set type.

For if the claim holds, then for every $\tau \in T(\Delta) \setminus BC(\Delta)$, τ is either $DBtype$ or a set type. That is, the first statement of the lemma holds. The second statement can also be

verified by *reductio*, using the claim. Suppose, for *reductio*, that there is $a \in R^G_{DBtype}$ and $\alpha \in Paths(\Delta)$ such that $a \neq r^G$ and $G \models \alpha(r^G, a)$. Then $\alpha \neq \epsilon$ since $a \neq r^G$. If α is not ϵ, then it follows from the claim that $type(\alpha)$ cannot be $DBtype$. However, if $G \models \Phi(\Delta)$ and $G \models \alpha(r^G, a)$, then by Lemma 10.6 and $a \in R^G_{DBtype}$, we have $type(\alpha) = DBtype$. This contradicts the claim.

Next, we show the claim by induction on $|\alpha|$.

Base case: $\alpha = K$ for some $K \in E$. By Definition 10.2, $type(K)$ must be a set type. Hence the claim holds in this case.

Inductive step: Assume the claim for $|\alpha|$. We show that the claim also holds for $\alpha \cdot K$, where $\alpha \cdot K \in Paths(\Delta)$. By the induction hypothesis, $type(\alpha)$ is either in $BC(\Delta)$ or is a set type. Since $\alpha \cdot K \in Paths(\Delta)$, $type(\alpha) \notin \mathcal{B}$. By Definition 10.2 and the definition of $T(\Delta)$, if $type(\alpha) \in \mathcal{C}$ then $type(\alpha \cdot K)$ is either in $BC(\Delta)$ or is a set type. Similarly, if $type(\alpha)$ is a set type, then $type(\alpha \cdot K) \in BC(\Delta)$. Hence the claim holds for $\alpha \cdot K$. ∎

Corollary 10.9: Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If there is G in $\mathcal{U}(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$, then there is H in $\mathcal{U}(\Delta)$ such that $R^H_{DBtype} = \{r^H\}$, $H \models \bigwedge \Sigma \wedge \neg\varphi$, and the size of H is no larger than the size of G. ∎

Proof: Let H be the substructure of G such that

$$|H| = \{a \mid a \in |G|, G \models \alpha(r^G, a) \text{ for some } \alpha \in Paths(\Delta)\}.$$

It is easy to verify that $H \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$ and

$$LB(H, \Sigma \cup \{\varphi\}) \;=\; LB(G, \Sigma \cup \{\varphi\}).$$

Hence by Lemma 10.5, $H \models \bigwedge \Sigma \wedge \neg\varphi$. By Lemma 10.8, we also have $R^H_{DBtype} = \{r^H\}$. ∎

In the sequel we assume that for every $\sigma(\Delta)$-structure G, $R^G_{DBtype} = \{r^G\}$. By Corollary 10.9, this assumption does not affect the outcome of word constraint implication in the context of $\mathcal{M}^{\approx}$.

Next, we define the identifying operation.

Definition 10.6: Let Δ be a schema in $\mathcal{M}^{\approx}$, G be a $\sigma(\Delta)$-structure that satisfies $\Phi(\Delta)$, $G = (|G|, r^G, E^G, R^G, Q^G)$, and o_1, o_2 be two distinct nodes in $|G|$. Then o_1 and o_2 are said to be *identifiable* if there is $\tau \in T(\Delta) \setminus BC(\Delta)$, such that $G \models R^G_\tau(o_1) \wedge R^G_\tau(o_2)$, and in addition, the following conditions are satisfied.

- If $\tau = \{\tau'\}$, then for every $a \in |G|$, $G \models *(o_1, a)$ iff $G \models *(o_2, a)$.
- If $\tau = [l_1 : \tau_1, \ \ldots, \ l_n : \tau_n]$, then for any $i \in [1, n]$ and $a \in |G|$, $G \models l_i(o_1, a)$ iff $G \models l_i(o_2, a)$.

The *structure resulting from identifying o_1 and o_2 in G* is defined to be

$$G_1 = (|G_1|, r^{G_1}, E^{G_1}, R^{G_1}, Q^{G_1}),$$

where

- $|G_1| = (|G| \setminus \{o_1, o_2\}) \cup \{o\}$, where $o \notin |G|$,
- $r^{G_1} = r^G$,
- for all $a, b \in |G_1|$ and $K \in E(\Delta)$,
  - if $a \neq o$ and $b \neq o$, then $G_1 \models K(a, b)$ iff $G \models K(a, b)$,
  - if $a = o$ and $b \neq o$, then $G_1 \models K(o, b)$ iff $G \models K(o_1, b)$ (iff $G \models K(o_2, b)$),
  - if $a \neq o$ and $b = o$, then $G_1 \models K(a, o)$ iff $G \models K(a, o_1) \vee K(a, o_2)$;
- for every $\tau' \in T(\Delta) \setminus \{\tau\}$, $R^{G_1}_{\tau'} = R^G_{\tau'}$; and in addition,

$$R^{G_1}_\tau = R^G_\tau \cup \{o\} \setminus \{o_1, o_2\};$$

- for every $\tau' \in T(\Delta) \setminus \{\tau\}$ and all $a, b \in |G_1|$, $a \approx^{G_1}_{\tau'} b$ iff $a \approx^G_{\tau'} b$; and in addition,

$$a \approx^{G_1}_\tau b \quad \text{iff} \quad (a, b) \text{ is in } \approx^G_\tau [o_1/o, o_2/o] \cup \{(o, o)\},$$

  where $\approx^G_\tau [o_1/o, o_2/o]$ stands for substituting o for every occurrence of o_1 and o_2 in the relation $\approx^G_\tau$.

■

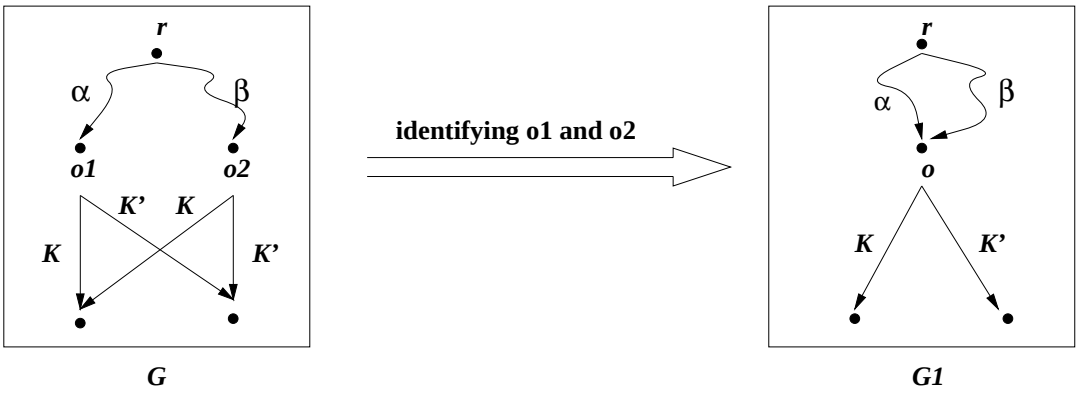

Figure 10.2: The identifying operation

The identifying operation is shown in Figure 10.2.

The following should be noted about structure G_1 described in Definition 10.6.

- The root r^G is not in $\{o_1, o_2\}$. This is because we assume $R^G_{DBtype} = \{r^G\}$ by Corollary 10.9.
- There is no $K \in E(\Delta)$ such that $G_1 \models K(o, o)$. This is because τ is not a class type. As a result, τ cannot be defined in terms of itself. Thus $G \not\models K(o_1, o_2) \vee K(o_2, o_1)$, since $G \models \Phi(\Delta)$.
- For every $a \in |G|$ and $K \in E(\Delta)$, $G_1 \models K(o, a)$ iff $G \models K(o_2, a)$. This is because $G \models K(o_1, a)$ iff $G \models K(o_2, a)$.

Next, we show that the identifying operation has certain properties.

Lemma 10.10: Let Δ, G, o_1, o_2, G_1 and o be as described in Definition 10.6. Then G_1 has the following properties.

1. $G_1 \models \Phi(\Delta)$.
2. For every $\alpha \in Paths(\Delta)$ and $a \in |G_1|$,
   (a) if $a \neq o$, then $a \in |G|$, and in addition, $G_1 \models \alpha(r^{G_1}, a)$ iff $G \models \alpha(r^G, a)$;
   (b) if $a = o$, then $G_1 \models \alpha(r^{G_1}, o)$ iff $G \models \alpha(r^G, o_1) \vee \alpha(r^G, o_2)$.

■

Proof: The first statement of the lemma can be easily verified by *reductio*. We prove the second statement by induction on $|\alpha|$.

Base case: $\alpha = \epsilon$. Note that $r^G \notin \{o_1, o_2\}$. Hence $G_1 \models \epsilon(r^{G_1}, a)$ iff $a = r^{G_1}$ iff $a = r^G$ iff $G \models \epsilon(r^G, a)$. Hence the statement holds for this case.

Inductive step: Assume the statement for $|\alpha|$. We show that the statement also holds for $\alpha \cdot K$, where $\alpha \cdot K \in Paths(\Delta)$.

(1) Assume that $a \neq o$. By Definition 10.6, clearly $a \in |G|$.

First, assume that $G \models \alpha \cdot K(r^G, a)$. Then there exists $b \in |G|$, such that

$$G \models \alpha(r^G, b) \wedge K(b, a).$$

If $b \notin \{o_1, o_2\}$, then by Definition 10.6, $b \in |G_1|$ and $b \neq o$. Thus by the induction hypothesis and Definition 10.6, we have $G_1 \models \alpha(r^{G_1}, b) \wedge K(b, a)$.

If $b \in \{o_1, o_2\}$, then by the induction hypothesis we have

$$G_1 \models \alpha(r^{G_1}, o).$$

By Definition 10.6, we have

$$G_1 \models K(o, a).$$

Hence $G_1 \models \alpha \cdot K(a, b)$.

Conversely, assume that $G_1 \models \alpha \cdot K(r^{G_1}, a)$. Then there exists $b \in |G_1|$ such that

$$G_1 \models \alpha(r^{G_1}, b) \wedge K(b, a).$$

If $b \neq o$, then by the induction hypothesis, $b \in |G|$ and

$$G \models \alpha(r^G, b).$$

By Definition 10.6, we have

$$G \models K(b, a).$$

Hence $G \models \alpha(r^G, b) \wedge K(b, a)$.

If $b = o$, then by the induction hypothesis,

$$G \models \alpha(r^G, o_1) \vee \alpha(r^G, o_2).$$

By Definition 10.6, we have

$$G \models K(o_1, a) \wedge K(o_2, a).$$

Hence $G \models \alpha(r^G, b) \wedge K(b, a)$.

(2) Assume that $a = o$.

First, assume that $G \models \alpha \cdot K(r^G, o_1) \vee \alpha \cdot K(r^G, o_2)$. Without loss of generality, assume that $G \models \alpha \cdot K(r^G, o_1)$. Then there exists $b \in |G|$, such that

$$G \models \alpha(r^G, b) \wedge K(b, o_1).$$

Clearly, $b \notin \{o_1, o_2\}$, since otherwise we would have had $G_1 \models K(o, o)$. By Definition 10.6, $b \in |G_1|$ and $b \neq o$. By the induction hypothesis,

$$G_1 \models \alpha(r^{G_1}, b).$$

By Definition 10.6, we have

$$G_1 \models K(b, o).$$

Hence $G_1 \models \alpha \cdot K(r^{G_1}, a)$.

Conversely, assume that $G_1 \models \alpha \cdot K(r^{G_1}, a)$. Then there exists $b \in |G_1|$ such that

$$G_1 \models \alpha(r^{G_1}, b) \wedge K(b, a).$$

Clearly $b \neq o$ since otherwise we would have had $G_1 \models K(o, o)$. By the induction hypothesis, $b \in |G|$ and

$$G \models \alpha(r^G, b).$$

By Definition 10.6,

$$G \models K(b, o_1) \vee K(b, o_2).$$

Hence $G \models \alpha \cdot K(r^G, o_1) \vee \alpha \cdot K(r^G, o_1)$.

This completes the proof of Lemma 10.10. ∎

Using Lemma 10.10, we study the impact of the identifying operation on word constraint implication. To do this, we first introduce the following notion.

Definition 10.7: Let Δ, G, o_1 and o_2 be as described in Definition 10.6. Let φ be a word constraint in $P_w(\Delta)$. Then G is said to *respect* $\neg\varphi$ *when identifying* o_1 *and* o_2 if there exists $a \in |G|$ such that

$$G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a),$$

and in addition, either $a \notin \{o_1, o_2\}$, or $a = o_1$ and $G \models \neg rt(\varphi)(r^G, o_2)$. ∎

Corollary 10.11: Let Δ, G, o_1, o_2 and G_1 be as described in Definition 10.6. Let $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If $G \models \Sigma$, then $G_1 \models \Sigma$. In addition, if G respects $\neg\varphi$ when identifying o_1 and o_2, then $G_1 \models \neg\varphi$. ∎

Proof: We first show that if $G \models \Sigma$, then $G_1 \models \Sigma$. Suppose, for *reductio*, that there exists $\phi \in \Sigma$ and $b \in |G_1|$, such that

$$G_1 \models lt(\phi)(r^{G_1}, b) \wedge \neg rt(\phi)(r^{G_1}, b).$$

If $b \neq o$, then by Lemma 10.10, $b \in |G|$ and in addition,

$$G \models lt(\phi)(r^G, b) \wedge \neg rt(\phi)(r^G, b).$$

This contradicts the assumption that $G \models \Sigma$.

If $b = o$, then by $G \models \phi$, we have

$$\begin{aligned} G &\models \neg lt(\phi)(r^G, o_1) \vee rt(\phi)(r^G, o_1), \\ G &\models \neg lt(\phi)(r^G, o_2) \vee rt(\phi)(r^G, o_2). \end{aligned}$$

Again by Lemma 10.10, we have

$$G_1 \models \neg lt(\phi)(r^G, o) \vee rt(\phi)(r^G, o).$$

This contradicts the assumption that $G_1 \not\models \phi$.

Next, we show that if G respects $\neg\varphi$ when identifying o_1 and o_2, then $G_1 \not\models \varphi$. Let $a \in |G|$ such that

$$G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a).$$

If $a \notin \{o_1, o_2\}$, then $a \in |G_1|$ and $a \neq o$. Since $G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a)$, by Lemma 10.10, we have

$$G_1 \models lt(\varphi)(r^{G_1}, a) \wedge \neg rt(\varphi)(r^{G_1}, a).$$

That is, $G_1 \not\models \varphi$.

If $a = o_1$ and $G \models \neg rt(\varphi)(r^G, o_2)$, then again by Lemma 10.10, we have

$$G_1 \models lt(\varphi)(r^{G_1}, o) \wedge \neg rt(\varphi)(r^{G_1}, o).$$

Therefore, $G_1 \not\models \varphi$. ∎

Next, we illustrate how to construct structures that satisfy equality constraints using the identifying operation.

Definition 10.8: Let Δ be a schema in $\mathcal{M}^{\approx}$ and H be a finite $\sigma(\Delta)$-structure satisfying $\Phi(\Delta)$. Then the *ultimately identified structure constructed from* H is defined to be structure G_m, where

$$\begin{aligned} G_1 &= H, \\ G_{i+1} &= \text{the structure resulting from identifying two identifiable nodes in } G_i, \end{aligned}$$

and G_m is the structure constructed at stage m such that there are no distinct identifiable nodes o_1, o_2 in $|G_m|$. ∎

Ultimately identified structures have the following properties.

Proposition 10.12: Let Δ and H be as described in Definition 10.8. Then the following statements hold.

1. The ultimately identified structure G constructed from H exists. In addition, the size of G is no larger than the size of H.

2. For every finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, if $H \models \Sigma$ then $G \models \Sigma$. In addition, if $H \models \neg\varphi$ and at each stage i of the construction described in Definition 10.8, G_i respects $\neg\varphi$ when identifying nodes, then $G \models \neg\varphi$.

3. $G \models \Phi(\Delta)$.

4. If for any $\tau \in T(\Delta)$, $H \models \forall x\, y\, (x \approx_\tau^H y \rightarrow x = y)$, then $G \models \Phi^{\approx}(\Delta)$.

■

Proof:

(1) To see that the ultimately identified structure G constructed from H exists, consider the sequence of structures constructed in Definition 10.8. This sequence is finite since H is finite and in addition, the size of G_{i+1} is strictly less than the size of G_i unless $i = m$. As a result, G exists. In addition, the size of G is no larger than the size of H.

(2) The second statement of the proposition can be verified by a straightforward induction on stage i, using Corollary 10.11.

(3) The third statement of the proposition follows from Lemma 10.10.

(4) To show the last statement of the proposition, first, by induction on i, it is easy to show that for any $\tau \in T(\Delta)$, $G_i \models \forall x\, y\, (x \approx_\tau^{G_i} y \rightarrow x = y)$. Therefore,

$$G \models \forall x\, y\, (x \approx_\tau^G y \rightarrow x = y). \qquad (\dagger)$$

Second, we show that $G \models \bigwedge_{\tau \in T(\Delta)} \forall x\, y\, \phi_\tau^{\approx}(x, y)$. Suppose, for *reductio*, that there exist $a, b \in |G|$ and $\tau \in T(\Delta)$, such that

$$G \models R_\tau^G(a) \wedge R_\tau^G(b) \wedge \neg\phi_\tau^{\approx}(a, b).$$

We consider the following cases of τ.

If $\tau = b$, or for some $C \in \mathcal{C}$, $\tau = C$, then by the definition of G and Definition 10.6, it is easy to see that

$$a \approx_\tau^G b \quad \text{iff } a = b.$$

That is, $G \models \phi^{\approx}_{\tau}(a,b)$. This contradicts the assumption.

If $\tau = \{\tau'\}$, then by (†) and $G \models \neg\phi^{\approx}_{\tau}(a,b)$, we have $G \not\models a \approx^G_{\tau} b$ and

$$G \models \forall x \exists y\,(*(a,x) \rightarrow *(b,y) \wedge x \approx^G_{\tau'} y) \wedge \forall x \exists y\,(*(b,x) \rightarrow *(a,y) \wedge x \approx^G_{\tau'} y).$$

By (†), we have $G \models a \neq b$ and

$$G \models \forall x\, y\,(x \approx^G_{\tau'} y \rightarrow x = y).$$

Therefore,

$$G \models \forall x\,(*(a,x) \leftrightarrow *(b,x)).$$

That is, a and b are identifiable. This contradicts the definition of G.

Similarly, if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then by (†) and $G \models \neg\phi^{\approx}_{\tau}(a,b)$, we have $G \not\models a \approx^G_{\tau} b$ and for every $i \in [1,n]$,

$$\forall x \forall y\,(l_i(a,x) \wedge l_i(b,y) \rightarrow x \approx^G_{\tau_i} y).$$

By (†), we have $G \models a \neq b$ and for every $i \in [1,m]$,

$$G \models \forall x\; y\,(x \approx^G_{\tau_i} y \rightarrow x = y).$$

Therefore, by $G \models \Phi(\Delta)$, for every $i \in [1,m]$, we have

$$G \models \forall x\,(l_i(a,x) \leftrightarrow l_i(b,x)).$$

That is, a and b are identifiable. This again contradicts the definition of G.

Therefore,

$$G \models \bigwedge_{\tau \in T(\Delta)} \forall x\, y\, \phi^{\approx}_{\tau}(x,y) \wedge \forall x\, y\,(\bigvee_{\tau \in T(\Delta)} x \approx_{\tau} y \rightarrow x = y).$$

That is, $G \models \Phi^{\approx}(\Delta)$.

This completes the proof of Proposition 10.12. ∎

10.3.3 The decidability of word constraint implication in $\mathcal{M}^{\approx}$

Finally, we show Theorem 10.3. More specifically, let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. We show that if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a small model G of size at most 2^{mN} and $G \in \mathcal{U}_f(\Delta)$, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ.

Consider $\tau = type(lt(\varphi))$. By Lemma 10.8, τ can be one of the following:

1. $\tau \in BC(\Delta)$,
2. $\tau = DBtype$,
3. $\tau = \{\tau'\}$.

For the first case, the filtration argument and identifying operation given above are sufficient to establish the small model property.

Corollary 10.13: Let Δ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$. If $type(lt(\varphi)) \in BC(\Delta)$ and $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model G in $\mathcal{U}_f(\Delta)$ such that the size of G is at most 2^{mN}, where m and N are as described in Proposition 10.7.

■

Proof: By Proposition 10.7, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a structure H such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta),$$

and the the size of H is at most 2^{mN}.

Let G be the ultimately identified structure constructed from H. By Proposition 10.12, the size of G is at most 2^{mN}. By the definition of H given in the proof of Proposition 10.7, we have that for every $\tau \in T(\Delta)$, $H \models \forall x\, y\, (x \approx_\tau^H y \rightarrow x = y)$. Hence again by Proposition 10.12, $G \models \Phi^{\approx}(\Delta)$. In addition, by $H \models \Phi(\Delta)$, we also have $G \models \Phi(\Delta)$.

Since $H \models \Sigma$, by Proposition 10.12, $G \models \Sigma$. In addition, by $H \models \neg\varphi$, there is $a \in |H|$ such that

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a).$$

By Lemma 10.6, $a \in R^H_\tau$ where $\tau = type(lt(\varphi))$. Since $\tau \in BC(\Delta)$, by Definition 10.6, it can be shown by a straightforward induction on i that a is in $|G_i|$ for every $i \in [1, m]$, where G_i is the structure constructed at stage i in the definition of G, as described in Definition 10.8. Indeed, a cannot be identified with any other node in any $|G_i|$. Hence for every $i \in [1, m]$, G_i respects $\neg\varphi$. Therefore, by Proposition 10.12, $G \models \neg\varphi$. ∎

Now assume that $\tau = DBtype$. Let H be the structure constructed in the proof of Proposition 10.7, and G be the ultimately identified structure constructed from H. By Corollary 10.9 and the proof of Proposition 10.7, it is easy to see that $R^H_{DBtype} = \{r^H\}$. Therefore, r^H cannot be identified with any other node in the construction of G. Thus by Proposition 10.12, G is indeed the small model of $\bigwedge \Sigma \wedge \neg\varphi$ described above.

Finally, let us consider $\tau = \{\tau'\}$. Let H and G be as described above, and G_i be the structure constructed at stage i in the definition of G. Consider node a in $|G_i|$ such that

$$G_i \models lt(\varphi)(r^{G_i}, a) \wedge \neg rt(\varphi)(r^{G_i}, a),$$

It is possible that when constructing G_{i+1}, a is identified with some $b \in |G_i|$ such that $H \models rt(\varphi)(r^{G_i}, b)$. If this happens, then it is possible that $G \models \varphi$.

To prevent this, we need to use a slightly stronger filtration argument. We first give the following definition and lemma, which illustrate the motivation for strengthening the argument given in the proof of Proposition 10.7.

Definition 10.9: Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$, φ be a word constraint in $P_w(\Delta)$ such that $type(lt(\varphi)) = \{\tau'\}$, and H be a $\sigma(\Delta)$-structure. Then H is said to satisfy the *isolation condition with respect to* φ if there exists $a \in |H|$ such that

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a),$$

and moreover, for any $b \in |H|$ such that $H \models rt(\varphi)(r^H, b)$, there exists $c \in |H|$ such that

$$\text{either } \; H \models *(a, c) \wedge \neg *(b, c) \; \text{ or } \; H \models \neg *(a, c) \wedge *(b, c).$$

∎

Lemma 10.14: Let Δ, H, G, Σ and φ be as described in Proposition 10.12. Assume that

$H \models \bigwedge \Sigma \wedge \neg\varphi$ and $type(lt(\varphi)) = \{\tau'\}$. Then $G \models \bigwedge \Sigma \wedge \neg\varphi$ if H satisfies the isolation condition with respect to φ. ∎

Proof: Since H satisfies the isolation condition with respect to φ, there exists $a \in |H|$, such that

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a),$$

and moreover, by Definition 10.6, for any $b \in |H|$ such that $H \models rt(\varphi)(r^H, b)$, a and b are not identifiable in H. Hence H respects $\neg\varphi$ when identifying nodes in H. In addition, the structure G_2 resulting from identifying any two identifiable nodes o_1 and o_2 in H also satisfies the isolation condition with respect to φ. To see this, first notice that by Definition 10.1, $lt(\varphi)$ must be of the form $\alpha \cdot K$, where K is a record label and either $type(\alpha) = DBtype$, or $type(\alpha) \in \mathcal{C}$. In both cases, for any node $o \in |H|$ such that $H \models \alpha(r^H, o)$, $o \notin \{o_1, o_2\}$. The same statement also holds for any $o \in |H|$ such that $H \models \alpha \cdot K \cdot *(r^H, o)$. Because of this, by Definitions 10.6 and 10.9, it is easy to verify that identifying o_1 and o_2 does not violate the isolation condition with respect to φ.

In fact, by a straightforward induction on i, it can be shown that every G_i constructed in the definition of G satisfies the isolation condition with respect to φ. Therefore, G_i respects $\neg\varphi$ when identifying identifiable nodes. Thus by Proposition 10.12, $G \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

The purpose of introducing a stronger filtration argument is to ensure the isolation condition. We give the argument in two steps. We first convert an arbitrary model of $\bigwedge \Sigma \wedge \neg\varphi$ to a model with certain property. We then refine the notion of *mlb* defined in the proof Proposition 10.7 based on this property.

Recall the notion of *lb* introduced in Section 10.3.1. For each $G \in \mathcal{U}(\Delta)$, we define an equivalence relation $\sim$ on $|G|$ as follows:

$$a \sim b \quad \text{iff} \quad lb(a, G, \Sigma \cup \{\varphi\}) = lb(b, G, \Sigma \cup \{\varphi\}).$$

Let $[o]$ denote the equivalence class of o with respect to $\sim$, and let $[G] = \{[o] \mid o \in |G|\}$.

Definition 10.10: Let Δ and φ be as described in Definition 10.9, G be a $\sigma(\Delta)$-structure,

and $a \in |G|$ such that

$$G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a).$$

Then G is said to have the *semi-isolation property with respect to* φ if it satisfies the following condition: for every $[b] \in [G]$ such that $rt(\varphi) \in lb(b, G, \Sigma \cup \{\varphi\})$, there exist $c, c' \in |G|$, such that one of the following holds:

- $G \models \neg * (a, c)$ and for any $o \in [b]$, $G \models *(o, c)$;
- $G \models *(a, c)$ and for any $o \in [b]$, $G \models \neg * (o, c)$;
- $G \models *(a, c) \wedge *(a, c')$, and there is a unique $b' \in [b]$ such that $G \models *(b', c) \wedge \neg *(b', c')$. In addition, for any $o \in [b]$, if $o \neq b'$, then $G \models \neg * (o, c)$.

The nodes c and c' are called the *isolating nodes for* a *and* $[b]$. ∎

Lemma 10.15: Let $\Delta = (\mathcal{C}, \nu, DBtype)$ be a schema in $\mathcal{M}^{\approx}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_w(\Delta)$ such that $type(lt(\varphi)) = \{\tau'\}$. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a $\sigma(\Delta)$-structure H such that $H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$ and H has the semi-isolation property with respect to φ. ∎

Proof: Let G be a model of $\bigwedge \Sigma \wedge \neg\varphi$ in $\mathcal{U}(\Delta)$. Then there exists $a \in |G|$ such that

$$G \models lt(\varphi)(r^G, a) \wedge \neg rt(\varphi)(r^G, a).$$

By $G \models \Phi(\Delta) \wedge \Phi^{\approx}(\Delta)$, for every $b \in |G|$ such that $G \models rt(\varphi)(r^G, b)$, there exists $c \in |G|$ such that either $G \models *(a, c) \wedge \neg * (b, c)$, or $G \models \neg * (a, c) \wedge *(b, c)$. For each $[b] \in [G]$ such that $rt(\varphi) \in lb(b, G, \Sigma \cup \{\varphi\})$, we modify G as follows.

- If there exists $c \in |G|$ and $b' \in [b]$ such that $G \models \neg * (a, c) \wedge *(b', c)$, then for all $o \in [b]$, we add an edge labeled with $*$ from o to c.
- If there exists $c \in |G|$ such that $G \models *(a, c)$ and for any $o \in [b]$, $G \models \neg * (o, c)$, then c is the isolating node for a and $[b]$.

- Otherwise there must be $c, c' \in |G|$ and $b' \in [b]$, such that $G \models *(a,\, c) \wedge *(a,\, c')$ and $G \models *(b',\, c) \wedge \neg * (b',\, c')$. In this case, for any $o \in [b]$, if $o \neq b'$, then we remove all the edges labeled with $*$ from o to c.

Let H be the structure resulting from modifying G as above. Then it is easy to see that H has the semi-isolation property with respect to φ and $H \models \Phi(\Delta)$. In addition, by Lemma 10.5, it is easy to verify that $H \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

Using Lemma 10.15, we present a stronger filtration argument as follows.

Proposition 10.16: Let Δ, Σ and φ be as described in Lemma 10.15. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model H such that

$$H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta),$$

H satisfies the isolation condition with respect to φ, and the size of H is at most 2^{mN}, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ. ∎

Proof: By Lemma 10.15, there exists a model G such that $G \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$, and G has the semi-isolation property with respect to φ. Hence there exists $a \in |G|$ such that

$$G \models lt(\varphi)(r^G,\, a) \wedge \neg rt(\varphi)(r^G,\, a),$$

and for every $b \in |G|$ such that $G \models rt(\varphi)(r^G,\, b)$, there exist isolating nodes $c_{[b]}, c'_{[b]}$ for a and $[b]$. Recall the notions of mlb and f given in the proof of Proposition 10.7. We refine the definition of mlb such that

- for every $b \in |G|$ such that $G \models rt(\varphi)(r^G,\, b)$,

  $$mlb(b) = (lb(b,\, G,\, \Sigma \cup \{\varphi\}),\ tag(b,\, c_{[b]}),\ tag(b,\, c'_{[b]})),$$

  where

  $$tag(b,\, o) = \begin{cases} \text{true} & \text{if } G \models *(b,\, o) \\ \text{false} & \text{otherwise} \end{cases}$$

- $mlb(a) = a$,

- for every $b \in |G|$ such that $G \models rt(\varphi)(r^G, b)$, $mlb(c_{[b]}) = c_{[b]}$ and $mlb(c'_{[b]}) = c'_{[b]}$.

Define function f and structure H as in the proof of Proposition 10.7. Similarly, it can be shown that $H \models \Phi(\Delta)$. Moreover, since $m \geq 2$, it is easy to verify that the size of H is at most 2^{mN}. In addition, for every $o \in |G|$, it can be verified that

$$lb(o, G, \Sigma \cup \{\varphi\}) \;=\; lb(f(o), H, \Sigma \cup \{\varphi\}). \qquad (\ddagger)$$

Hence by Lemma 10.5 and the assumption that $G \models \bigwedge \Sigma \wedge \neg\varphi$, we have $H \models \bigwedge \Sigma \wedge \neg\varphi$.

Next, we show that H satisfies the isolation condition with respect to φ.

By the definition of mlb, for any $b \in |G|$ such that $G \models rt(\varphi)(r^G, b)$, $f(c_{[b]}) = c_{[b]}$ and $f(c'_{[b]}) = c'_{[b]}$. We also have $f(a) = a$. In addition, by the the definition of H, it is easy to show the following:

$$\begin{aligned}
H \models *(f(b), c_{[b]}) \quad &\text{iff} \quad G \models *(b, c_{[b]}) \\
H \models *(f(b), c'_{[b]}) \quad &\text{iff} \quad G \models *(b, c'_{[b]}) \\
H \models *(a, c_{[b]}) \quad &\text{iff} \quad G \models *(a, c_{[b]}) \\
H \models *(a, c'_{[b]}) \quad &\text{iff} \quad G \models *(a, c'_{[b]})
\end{aligned}$$

Hence $c_{[b]}$ and $c'_{[b]}$ are isolating nodes for a and $[f(b)]$ in H. By $(\ddagger)$, we have

$$H \models lt(\varphi)(r^H, a) \wedge \neg rt(\varphi)(r^H, a).$$

Moreover, for any $o \in |H|$, if $H \models rt(\varphi)(r^H, o)$, then $o = f(b)$ for some $b \in |G|$ such that $G \models rt(\varphi)(r^G, b)$. Therefore, by Definitions 10.10 and 10.9, it can be shown that H satisfies the isolation condition with respect to φ. ∎

The following corollary completes the proof of Theorem 10.3.

Corollary 10.17: Let Δ, Σ and φ be as described in Lemma 10.15. If $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model H in $\mathcal{U}_f(\Delta)$ such that the size of H is at most 2^{mN}, where N is the length of $\bigwedge \Sigma \wedge \neg\varphi$, and m is the maximum record width of Δ. ∎

Proof: By Lemma 10.15, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then there is a $\sigma(\Delta)$-structure H such that $H \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$ and H has the semi-isolation property with respect to

φ. Thus by Proposition 10.16, there is H' such that $H' \models \bigwedge \Sigma \wedge \neg\varphi \wedge \Phi(\Delta)$, H' satisfies the isolation condition with respect to φ, and the size of H' is at most 2^{mN}. Let G be the ultimately identified structure constructed from H'. Then by Lemma 10.14,

$$G \models \bigwedge \Sigma \wedge \neg\varphi.$$

By Proposition 10.12, the size of G is at most 2^{mN}, and in addition,

$$G \models \Phi(D).$$

By the definition of H' given in the proof of Proposition 10.16 (see also the proof of Proposition 10.7), we have that for every $\tau \in T(\Delta)$, $H' \models \forall x\, y\, (x \approx_\tau^{H'} y \rightarrow x = y)$. Thus again by Proposition 10.12, we have

$$G \models \Phi^{\approx}(D).$$

Hence $G \in \mathcal{U}_f(\Delta)$. ∎

Part IV

Conclusion

This dissertation has introduced a path constraint language, P_c, and investigated its associated implication and finite implication problems in a variety of database contexts.

- In Part I, applications of path constraints of P_c have been described.
- In Part II, path constraint implication has been studied in the context of semistructured databases, i.e., databases without schemas.
- Part III, path constraint implication has been investigated in the context of structured databases, i.e., databases with schemas.
- A number of complexity results on path constraint implication have been established in Parts II and III. These results are summarized in Part IV. In addition, directions for further research are also identified in this part.

Chapter 11

Conclusion and Further Work

This chapter summarizes the results reported in this dissertation and identifies further research directions.

11.1 Summary

The primary goal of this research has been to explore the applications of path (inclusion) constraints and to settle the question of path constraint implication in a variety of database contexts.

There is a natural analogy between this work and inclusion dependency theory developed for the relational model. Inclusion dependency theory has been well established as an important part of relational database theory (see [5] for a survey of dependency theory). It has proven very useful for relational databases. However, with the development of more sophisticated database models with richer constructs than the relational model, dependency (constraint) theory has been considered somewhat out of fashion. Little work on inclusion constraint theory for the new data models has been seen. In fact, to my knowledge, only the work reported in [9] is close to this research.

This dissertation argues that path (inclusion) constraints can play an important role in a variety of database contexts. To this end, a path constraint language, P_c, has been introduced. This constraint language captures a number of natural integrity constraints. Among these constraints are extent constraints, inverse relationships and local database constraints, which are commonly found in object-oriented databases. These constraints are not only a fundamental part of the semantics of the data, but are also important in query optimization. Recently, they have gained a new vigor in semistructured databases with the popularity of the World-Wide Web. In particular, these constraints can be used

to specify structural information of semistructured data, which is not constrained by a schema, and can even be used to support some object-oriented features. In the context of sophisticated database models such as object-oriented models, these constraints also arise from a wide range of applications. They are useful not only for query optimization, but also for database updates and transformations. Applications of the path constraints of P_c have been addressed in Part I.

The question of logical implication for path constraints is the most important theoretical question in connection with path constraints. To take advantage of path constraints, it is important to be able to reason about them. Implication and finite implication of path constraints are not only interesting in their own right, but are also related to problems in finite model theory, bounded variable logics, feature logics, logics of programs and rewrite systems. The focus of this dissertation has been on path constraint implication.

Complexity results. The implication and finite implication problems for path constraints of P_c have been studied in Part II for semistructured databases. Two semistructured data models have been considered. One is the semistructured data model [2, 8, 18, 20, 68, 70], referred to as $\mathcal{SM}$. In $\mathcal{SM}$, data is represented as a rooted, edge-labeled, directed graph. The other is the deterministic data model recently developed in [21], referred to as $\mathcal{DM}$. The model $\mathcal{DM}$ is a variant of $\mathcal{SM}$, in which data is represented as a rooted, edge-labeled, directed graph with deterministic edge relations. That is, the edges emanating from any node in the graph have distinct labels. It turns out that path constraint implication has widely different complexities in these two models. In the context of $\mathcal{DM}$, the implication and finite implication of constraints of P_c are simple enough to be decidable in cubic-time and finitely axiomatizable. In contrast, in the context of $\mathcal{SM}$, these problems become hard enough to be undecidable.

In the context of $\mathcal{SM}$, it has been shown that despite the simple syntax of the language P_c, its associated implication problem is r.e. complete and its finite implication problem is co-r.e. complete. These undecidability results also hold for two fragments of P_c. One of the fragments is the largest subset of P_c without equality. The other is the set of path constraints of P_c having the forward form. Indeed, the existence of a conservative reduction

	Implication	*Finite implication*
Path constraint language P_c	r.e. complete	co-r.e. complete
Sublanguage P_f *of* P_c	r.e. complete	co-r.e. complete
Sublanguage P_+ *of* P_c	r.e. complete	co-r.e. complete
Prefix restricted subsets of P_c	Decidable	Decidable
Sublanguage P_β *of* P_c	Decidable	Decidable
Prefix extended subsets of P_β	Decidable	Decidable

Table 11.1: Path constraint implication in the context of semistructured data ($\mathcal{SM}$)

from the set of all first-order sentences to each of the two fragments has been established. These undecidability results are rather surprising since P_c is a mild generalization of the class of word constraints, P_w, introduced and studied in [9], which possesses decidable implication and finite implication problems.

However, there is good news. The undecidability results motivated the search for decidable fragments of the language P_c which retain sufficient expressive power. Several fragments have been identified, which share the following properties. First, they each properly contain the set of word constraints. Second, in contrast to the class of word constraints, each of these fragments fails to be included in two-variable first-order logic, a fragment of first-order logic whose decidability is known. Third, they allow the formulation of many semantic relations which are of interest from the point of view of database theory, including extent constraints, inverse relationships and local database constraints commonly found in object-oriented databases. And finally, they each possess decidable implication and finite implication problems.

In addition, a generalization of P_c, $P_c^\wedge$, has been investigated. It has been shown that the undecidability and decidability results on the fragments of P_c also hold on the analogous fragments of $P_c^\wedge$.

The main results on path constraint implication in the context of $\mathcal{SM}$ are summarized in Table 11.1.

	Implication	*Finite implication*
Word constraint language P_w	Decidable (cubic-time) Finitely axiomatizable	Decidable (cubic-time) Finitely axiomatizable
Path constraint language P_c	Decidable (cubic-time) Finitely axiomatizable	Decidable (cubic-time) Finitely axiomatizable
Extension P_c^- *of* P_c	Decidable	Decidable
Extension P_c^* *of* P_c	Undecidable	Undecidable

Table 11.2: Path constraint implication in the context of semistructured data ($\mathcal{DM}$)

In the context of $\mathcal{DM}$, four path constraint languages have been considered: P_w, P_c, P_c^- and P_c^*. The constraint languages P_c^- and P_c^* generalize P_c by including wildcards in path expressions and by representing paths as regular expressions, respectively. In contrast to the PTIME decidability of word constraint implication established in the context of $\mathcal{SM}$ by [9], it has been shown that in $\mathcal{DM}$, the implication and finite implication problems for P_w are decidable in cubic-time. In contrast to the undecidability results in $\mathcal{SM}$ mentioned above, the implication and finite implication problems for P_c in $\mathcal{DM}$ are not only decidable in cubic-time, but are also finitely axiomatizable. In addition, the implication and finite implication problems for P_c^- are also decidable in $\mathcal{DM}$. These demonstrate that the determinism condition of $\mathcal{DM}$ simplifies reasoning about path constraints. However, the implication and finite implication problems for P_c^* remain undecidable in $\mathcal{DM}$. This undecidability result shows that the determinism condition of $\mathcal{DM}$ does not reduce the analysis of path constraint implication to a trivial problem. See Table 11.2 for the main results established in the context of $\mathcal{DM}$.

It is tempting to directly apply the complexity results established for semistructured data to structured data, i.e., data constrained by a schema. However, the presence of type systems or schemas can alter the computational complexity of the path constraint implication problem in unexpected ways. The type system or schema definition can be viewed as imposing a type constraint on the data. The type constraints give rise to a host of new problems.

	Object-oriented data models		
	$\mathcal{M}$	$\mathcal{M}^+$	$\mathcal{M}_f^+$
Implication and finite implication for P_w	Decidable (cubic-time) Finitely axiomatizable	Decidable	Decidable
Implication and finite implication for P_c	Decidable (cubic-time) Finitely axiomatizable	Undecidable	Undecidable

Table 11.3: Path constraint implication in the context of structured data

Path constraint implication has been investigated for structured data in Part III. To explore the impacts of different type constructs on path constraint implication, three practical object-oriented data models have been considered: $\mathcal{M}$, $\mathcal{M}^+$ and $\mathcal{M}_f^+$. The model $\mathcal{M}$ supports classes, records and recursive structures. The models $\mathcal{M}^+$ and $\mathcal{M}_f^+$ extend $\mathcal{M}$ by also supporting sets and finite sets, respectively. These models are similar to those studied in [3, 5, 6, 31, 58]. The implication and finite implication problems for P_w and P_c have been investigated in the context of each of these models. The main results on these problems are shown in Table 11.3.

One might think that the imposition of a type system, which imposes some regularity on the data, would simplify the analysis of path constraint implication. This may be the case. However one of the main results of this dissertation is to establish the possibly surprising result that the presence of types actually *complicates* the implication problem for path constraints: there are decidable path constraint problems that become undecidable in the presence of types. It has been shown that adding a type constraint to the data may in some cases simplify reasoning about path constraints, and in other cases make it harder. Because of this, results on path constraint implication developed in the context of semistructured databases may no longer hold in the presence of types. To demonstrate this, two forms of implication problems associated with path constraints of P_c have been investigated. One is referred to as the extended implication and finite implication problems for P_w (EIPs). The other is referred to as the prefix bounded implication and finite implication problems for P_c (PBIPs). It has been shown that, on the one hand, EIPs are undecidable in the

	The extended (finite) implication for P_w	*The prefix bounded (finite) implication for P_c*
The semistructured data model $\mathcal{SM}$	Undecidable	Decidable (PTIME)
Object-oriented model $\mathcal{M}$	Decidable (cubic-time)	Decidable (cubic-time)
Object-oriented model $\mathcal{M}^+$	Undecidable	Undecidable
Object-oriented model $\mathcal{M}^+_f$	Undecidable	Undecidable

Table 11.4: The interaction between path and type constraints

context of the semistructured data model $\mathcal{SM}$, but they become decidable in cubic-time in the context of the object-oriented model $\mathcal{M}$. On the other hand, PBIPs are decidable in PTIME in $\mathcal{SM}$, but in the context of the object-oriented models $\mathcal{M}^+$ and $\mathcal{M}^+_f$, these decidability results break down. The interaction between path and type constraints is demonstrated in Table 11.4.

Another issue studied in Part III is the impact of complex value equality on path constraint implication. As in OEM [8, 68, 70], it is assumed that in $\mathcal{SM}$ and $\mathcal{DM}$, each data entity has a unique identity, and two data entities are equal if and only if they have the same identify. This equality relation is referred to as identity equality. In $\mathcal{M}$, $\mathcal{M}^+$ and $\mathcal{M}^+_f$, the equality relation is also defined to be identity equality. However, in some object-oriented database systems such as those studied in [5, 6, 31, 58], complex values with nested structures are common. Examples of such complex values are sets of records or records with set components. These complex values may not have a unique identity. As a result, the equality relation on complex values is defined to be complex value equality. It has been shown that complex value equality can be characterized by an equality constraint, which is definable in first-order logic. The interaction between equality and path constraints has been addressed. In particular, word constraint implication has been studied in the context of an object-oriented model $\mathcal{M}^{\approx}$, which supports complex values with nested structures.

An abstraction of databases of $\mathcal{M}^{\approx}$ has been presented in terms of equality and type constraints. In the presence of both equality and type constraints, it has been shown that the implication and finite implication problems for P_w remain decidable.

Proof techniques. A number of undecidability and decidability results have been established in this dissertation. Below we briefly summarize the techniques used in the proofs of these results.

The techniques for establishing the undecidability results include the following.

- Reduction from the halting problem for two-register machines [1, 15] (Theorems 4.1, 4.2 and 4.3).
- Reduction from the word problem for (finite) monoids [5, 62] (Theorems 6.19, 8.1, 8.13, 9.1, 9.2, 9.5, 9.6, 9.13 and 9.19).

The techniques for establishing the decidability results include the following.

- Small model argument [15] (Theorem 5.1, Propositions 6.4, 6.10 and 6.18).
- Filtration argument [72] (Theorems 5.7 and 10.3).
- Algorithm for testing implication (Corollaries 6.9, 6.17 and Theorem 8.17).
- Finite axiomatization for both implication and finite implication (Propositions 8.10 and 8.16). The idea is to develop a finite set of inference rules that are sound and complete for both implication and finite implication. As a result, the implication and finite implication problems coincide and are decidable.
- Reduction to decidable decision problems (Theorem 5.9, 8.2, 8.14 and 9.7).

11.2 Further research

There remain many open questions related to the work reported in this dissertation. Some immediate and significant extensions to this work are described below.

There are two questions about path constraint implication to which I have not yet been able to find satisfactory answers. One is about implication and finite implication of constraints of a sublanguage of P_c^*, P_w^*, in the context of the deterministic data model $\mathcal{DM}$. The language P_w^* is defined as follows:

$$P_w^* = \{\phi \mid \phi \in P_c^*,\ \phi \text{ is of the forward form, } pf(\phi) = \epsilon\}.$$

In the context of the semistructured data model $\mathcal{SM}$, P_w^* has been studied in [9]. It was shown there that in $\mathcal{SM}$, the implication and finite implication problems for P_w^* are decidable in EXPSPACE. In the context of $\mathcal{DM}$, it has been shown in this dissertation that the implication and finite implication problems for P_c^* are undecidable. However, it remains open whether the implication and finite implication problems for P_w^* are decidable in the context of $\mathcal{DM}$. These problems are closely related to the satisfiability and finite satisfiability problems for deterministic propositional dynamic logic with converse [49, 75], which are, to my knowledge, also open.

The other open question related to path constraint implication is the interaction between path constraints and complex value equality. In Chapter 10 of this dissertation, it has been shown that word constraint implication is decidable in the context of an object-oriented model $\mathcal{M}^{\approx}$, which supports complex values with nested structures. It remains open whether the implication and finite implication problems for P_c are decidable in the context of $\mathcal{M}^{\approx}$. A conjecture here is that these problems are undecidable, and the undecidability may be established by reduction from the word problem for (finite) monoids.

A significant extension to this work would be a study of how to check and maintain path constraints, which is left undone in this dissertation. The most promising approach to achieving this is perhaps by developing an incremental mechanism. In particular, the incremental maintenance algorithm for materialized views over semistructured data developed in [7] may shed light here.

Another significant extension would be a design of path constraint syntax that is conformable with XML and XML DTD [17]. For example, consider the following P_c constraints given in Chapter 1:

$$\forall\, x\,(book \cdot author(r,\, x) \rightarrow person(r,\, x))$$
$$\forall\, x\,(person \cdot wrote(r,\, x) \rightarrow book(r,\, x))$$

$$\forall\, x\,(book(r,\, x) \rightarrow \forall\, y\,(author(x,\, y) \rightarrow wrote(y, x)))$$
$$\forall\, x\,(person(r,\, x) \rightarrow \forall\, y\,(wrote(x,\, y) \rightarrow author(y,\, x)))$$

In XML syntax, these constraints may be expressed as:

```
<constraint>
     <inclusion path = "book.author"  memberOf ="person" />
</constraint>

<constraint>
     <inclusion path = "person.wrote"  memberOf ="book" />
</constraint>

<constraint>
     <prefix     path = "book" />
     <inverse    path = "author"  inverseOf = "wrote"/>
</constraint>

<constraint>
     <prefix     path = "person" />
     <inverse    path = "wrote"  inverseOf = "author"/>
<constraint>
```

One should be able to describe in this syntax external links such as those studied in [63]. To accomplish this, much more remains to be done.

Bibliography

[1] S. O. Aanderaa. "On the decision problem for formulas in which all disjunctions are binary". In *Proceedings of the 2nd Scandinavian Logic Symposium*, pp. 1-18, 1971.

[2] S. Abiteboul. "Querying semi-structured data". In *Proceedings of the 6th International Conference on Database Theory*, 1997.

[3] S. Abiteboul and J. Van den Bussche. "Deep equality revisited". In *Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases*, pp. 213-228, 1995.

[4] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, and J. Siméon. "Querying documents in object databases". *Journal of Digital Libraries*, 1(1), 1997.

[5] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesly Publishing Company, 1995.

[6] S. Abiteboul and P. C. Kanellakis. "Object identity as a query primitive". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 159-173, 1989.

[7] S. Abiteboul, J. McHugh, M. Rys, V. Vassalos, and J. Wiener. "Incremental maintenance for materialized views over semistructured data". In *Proceedings of International Conference on Very Large Databases*, 1998.

[8] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Weiner. "The lorel query language for semistructured data". *Journal of Digital Libraries*, 1(1), 1997.

[9] S. Abiteboul and V. Vianu. "Regular path queries with constraints", In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, 1997.

[10] S. Abiteboul and V. Vianu. "Queries and computation on the Web". In *Proceedings of the 6th International Conference on Database Theory*, pp. 262-275, 1997.

[11] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an object-oriented database system: the story of O2*. Morgan Kaufmann, San Mateo, California, 1992.

[12] J. Barwise. "On Moschovakis closure ordinals". *Journal of Symbolic Logic*, 42:292-296, 1977.

[13] C. Beeri. "A formal approach to object-oriented databases". *IEEE Transactions on Knowledge and Data Engineering*, 5: 353-382, 1990.

[14] M. F. van Bommel and G. E. Weddell. "Reasoning about equations and functional dependencies on complex objects". *IEEE Transactions on Knowledge and Data Engineering*, 6(3): 455-469, 1994.

[15] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Springer, 1997.

[16] T. Bray, C. Frankston, and A. Malhotra. "Document Content Description for XML". W3C Note NOTE-dcd-19980731. Available as `http://www.w3.org/TR/NOTE-dcd`.

[17] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. "Extensible Markup Language (XML) 1.0". W3C Recommendation REC-xml-19980210. Available as `http://www.w3.org/TR/REC-xml`.

[18] P. Buneman. "Semistructured data". Tutorial in *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, 1997.

[19] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. "Adding structure to unstructured data". In *Proceedings of the 6th International Conference on Database Theory*, 1997.

[20] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. "A query language and optimization techniques for unstructured data". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 505-516, 1996.

[21] P. Buneman, A. Deutsch, and W. Tan. "A deterministic model for semi-structured data". In *Proceedings of Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, 1999.

[22] P. Buneman, W. Fan, and S. Weinstein. "Path constraints on semistructured and structured data". In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 1998. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/pods98.ps.gz`.

[23] P. Buneman, W. Fan, and S. Weinstein. "Interaction between path and type constraints". In *Proceedings of the 18th ACM Symposium on Principles of Database Systems*, 1999. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/pods99.ps.gz`.

[24] P. Buneman, W. Fan, and S. Weinstein. "Path constraints in semistructured databases". To appear in *Journal of Computer and System Sciences*. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/jcss99.ps.gz`.

[25] P. Buneman, W. Fan, and S. Weinstein. "Some undecidable implication problems for path constraints". Technical Report MS-CIS-97-14, Department of Computer and Information Science, University of Pennsylvania, 1997. Available as `ftp://ftp.cis.upenn.edu/ pub/papers/db-research/tr9714.ps.gz`.

[26] P. Buneman, W. Fan, and S. Weinstein. "The decidability of some restricted implication problems for path constraints". Technical Report MS-CIS-97-15, Department of Computer and Information Science, University of Pennsylvania, 1997. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/tr9715.ps.gz`.

[27] P. Buneman, W. Fan, and S. Weinstein. "Path constraints in the presence of types". Technical Report MS-CIS-97-16, Department of Computer and Information Sci-

ence, University of Pennsylvania, 1997. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/tr9716.ps.gz`.

[28] P. Buneman, W. Fan, and S. Weinstein. "Interaction between path and type constraints". Technical Report MS-CIS-98-16, Department of Computer and Information Science, University of Pennsylvania, 1998. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/tr9816.ps.gz`.

[29] P. Buneman, W. Fan, and S. Weinstein. "Equality, type and word constraints". Technical Report MS-CIS-98-32, Department of Computer and Information Science, University of Pennsylvania, 1998. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/tr9832.ps.gz`.

[30] P. Buneman, W. Fan, and S. Weinstein. "Path constraints on deterministic graphs". Technical Report MS-CIS-98-33, Department of Computer and Information Science, University of Pennsylvania, 1998. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/tr9833.ps.gz`.

[31] R. G. G. Cattell, editor. *The object-oriented standard: ODMG-93* (Release 1.2). Morgan Kaufmann, San Mateo, California, 1996.

[32] U. S. Chakravarthy, J. Grant, and J. Minker. "Foundations of semantic query optimization for deductive databases". In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, San Mateo, California, 1988.

[33] V. Christophides, S. Cluet, and G. Moerkotte. "Evaluating queries with generalized path expressions". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1996.

[34] S. Cluet and C. Delobel. "A general framework for the optimization of object-oriented queries". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1992.

[35] N. Coburn and G. E. Weddell. "Path constraints for graph-based data model: Towards a unified theory of typing constraints, equations and functional dependencies". In

Proceedings of the 2nd International Conference on Deductive and Object-Oriented Databases, pp. 312-331, 1991.

[36] S. Cosmadakis, P. Kanellakis, and M. Vardi. "Polynomial-time implication problems for unary inclusion dependencies". *Journal of ACM*, 37(1): 15-46, January 1990.

[37] U. Dayal. "Queries and views in an object-oriented data model". In *Proceedings of the 2nd International Workshop on Database Programming Languages*, pp. 80-102, 1989.

[38] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "XML-QL: a query language for XML". W3C Note NOTE-xml-ql-19980819. Available as `http://www.w3.org/TR/NOTE-xml-ql`.

[39] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[40] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. "Catching the boat with Strudel: experience with a Web-site management system". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 414-425, 1998.

[41] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "A query language and processor for a web-site management system". In *Workshop on Management of Semistructured Data*, 1997.

[42] M. Fernandez and D. Suciu. "Query optimizations for semi-structured data using graph schemas". In *Proceedings of International Conference on Data Engineering*, 1998.

[43] D. Florescu, L. Raschid, and P. Valduriez. "A methodology for query reformulation in CIS using semantic knowledge". *Special issue on Formal Methods in Cooperative Information Systems*, 5(4), 1996.

[44] M. Fuchs, M. Maloney, and A. Milowski. "Schema for object-oriented XML". W3C Note NOTE-SOX-19980930. Available as `http://www.w3.org/TR/NOTE-SOX`.

[45] E. Grädel, P. Kolaitis, and M. Vardi. "On the decision problem for two-variable first-order logic". *Bulletin of Symbolic Logic*, 3(1): 53-69, March 1997.

[46] E. Grädel, M. Otto, and E. Rosen. "Two-variable logic with counting is decidable". In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, pp. 306-317, 1997.

[47] E. Grädel, M. Otto, and E. Rosen. "Undecidability results on two-variable logics". In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, *LNCS 1200*, 1997.

[48] A. G. Hamilton. *Logic for mathematicians*. Cambridge University Press, 1978.

[49] D. Harel. "Dynamic logic". In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic. II: Extensions of Classical Logic*, pp. 497-604, 1984.

[50] W. Hodges. *Model theory*. Cambridge University Press, 1993.

[51] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesly Publishing Company, 1979.

[52] N. Immerman. "Upper and lower bounds for first order expressibility". *Journal of Computer and System Sciences*, 25: 76-98, 1982.

[53] N. Immerman. "Languages that capture complexity classes". *SIAM Journal of Computing*, 16: 760-778, 1987.

[54] M. Ito and G. E. Weddell. "Implication problems for functional constraints on databases supporting complex objects". *Journal of Computer and System Sciences*, 50 (1): 165-187, 1995.

[55] M. Ito, G. E. Weddell, and N. Coburn. "On specialization constraints over complex objects". Technical Report CS-91-62, Department of Computer Science, University of Waterloo, 1991.

[56] M. Kifer, W. Kim, and Y. Sagiv. "Querying object-oriented databases". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 393-402, 1992.

[57] D. Konopnicki and O. Shmueli. "Draft of W3QS: a query system for the World-Wide Web". In *Proceedings of International Conference on Very Large Databases*, 1995.

[58] Anthony Kosky. *Transforming databases with recursive data structures*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.

[59] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. "The ObjectStore Database system". *Communications of the ACM*, 34(10): 51-63, October 1991.

[60] O. Lassila and R. R. Swick. "Resource Description Framework (RDF) model and syntax specification". W3C Working Draft WD-rdf-syntax-19981008. Available as `http://www.w3.org/TR/WD-rdf-syntax`.

[61] A. Layman, E. Jung, E. Maler, H. S. Thompson, J. Paoli, J. Tigue, N. H. Mikula, and S. De Rose. "XML-Data". W3C Note NOTE-XML-data-980105. Available as `http://www.w3.org/TR/1998/NOTE-XML-data`.

[62] H. R. Lewis and C. H. Papadimitriou. *Elements of the theory of computation*. Prentice-Hall, 1981.

[63] E. Maler and S. De Rose. "XML Linking language (XLink)". W3C Working Draft WD-xlink-19980303. Available as `http://www.w3.org/TR/WD-xlink`.

[64] E. Maler and S. De Rose. "XML Pointer language (XPintor)". W3C Working Draft WD-xptr-19980303. Available as `http://www.w3.org/TR/WD-xptr`.

[65] A. O. Mendelzon, G. A. Mihaila, and T. Milo. "Querying the World Wide Web". In *Proceedings of the 4th International Conference on Parallel and Distributed Information Systems*, pp. 80-91, 1996.

[66] J. Mylopoulos, P. Bernstein, and H. Wong. "A language facility for designing database-intensive applications". *ACM Transactions on Database Systems*, 5(2), June 1980.

[67] S. Nestorov, S. Abiteboul, and R. Motwani. "Inferring structure in semistructured data". In *Workshop on Management of Semistructured Data*, 1997.

[68] S. Nestorov, J. Ullman, J. Weiner, and S. Chawathe. "Representative objects: Concise representations of semistructured, hierarchical data". In *Proceedings of the 13th International Conference on Data Engineering*, 1997.

[69] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. "Object fusion in mediator systems". In *Proceedings of the 22nd International Conference on Very Large Databases*, pp. 413-424, 1996.

[70] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. "Object exchange across heterogeneous information sources". In *Proceedings of the 11th International Conference on Data Engineering*, pp. 251-260, March 1995.

[71] L. Popa and V. Tannen. "An equational chase for path-conjunctive queries, constraints, and views". In *Proceedings of the 7th International Conference on Database Theory*, 1999.

[72] W. C. Rounds. "Feature logics". In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, 1996.

[73] D. Suciu. "Query decomposition and view maintenance for query languages for unstructured data". In *Proceedings of International Conference on Vary Large Databases*, 1996.

[74] J. Thierry-Mieg and R. Durbin. "Syntactic definitions for the ACEDB data base manager". Technical Report MRC-LMB xx.92, MRC Laboratory for Molecular Biology, Cambridge, CB2 2QH, UK, 1992.

[75] M. Y. Vardi and P. Wolper. "Automata-theoretic techniques for modal logic of programs". *Journal of Computer and System Sciences*, 32(2): 182-221, April 1986.

[76] N. Walsh. "A guide to XML". In D. Connolly, editor, *XML: principles, tools and techniques*. O'Reilly, 1997.

[77] H. Wang. "Dominoes and the $\forall\exists\forall$-case of the decision problem". In *Proceedings of Symposium on Mathematical Theory of Automata*, Brooklyn Polytechnic Institute, pp. 23-55, 1962.

[78] G. E. Weddell. "Reasoning about functional dependencies generalized for semantic data models." *ACM Transactions on Database Systems*, 17(1): 32-64, March 1992.

[79] G. E. Weddell and N. Coburn. "A theory of specialization constraints for complex objects." In *Proceedings of the 3rd International Conference on Database Theory*, pp. 229-244, December 1990.

[80] C. Zaniolo. "The database language GEM". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 423-434, 1983.